COUPLED SENSOR CONFIGURATION AND PATH-PLANNING IN UNCERTAIN ENVIRONMENTS USING MULTIMODAL SENSORS

by
Chase St. Laurent

A dissertation submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN MECHANICAL ENGINEERING.

April 20, 2022

APPROVED:

Dr. Raghvendra V. Cowlagi, Advisor
Associate Professor, Aerospace Engineering Department.

Dr. Yihao Zheng, Graduate Committee Representative
Assistant Professor, Mechanical Engineering Department, WPI.

Dr. Zhi Li, Committee Member
Assistant Professor, Robotics Engineering Department, WPI.

Dr. Andrew Clark, Committee Member
Associate Professor, Electrical and Computer Engineering Department, WPI.

Dr. Dmitry Korkin, Committee Member
Professor, Computer Science Department, WPI.

Acknowledgements

To the reader who has decided to begin reading this dissertation, thank you for your interest and I hope you learn something new and valuable. To the reader who claims to make it to the very end, thank you, but flipping from the cover page to the last page doesn't count! To my dissertation committee who does in fact review this dissertation, I sincerely thank you for your time as well as the advice you have given me at various stages during my PhD.

As this dissertation will describe, sensor configuration and path-planning tasks are performed better when *coupled*. Throughout this PhD I came to realize this was very true about my relationship with my PhD advisor Raghvendra Cowlagi. While I may have been the sensor network, collecting data and formulating the methods, the mission could not have been a success without the careful planning and guidance I received from Raghu.

Speaking of the method in this dissertation, I cannot go without recognizing my peers Ben Cooper and Jie Fang who did extensive related work in the interactive planning and sensing field. Without their dedication and innovation, this work may not have been made possible. A special mention to Andrea Gjokollari for helping with the most tedious of tasks during experimentation.

I am grateful for the financial support of the Mechanical Engineering Department and for the valuable lessons my Teaching Assistant position gave me. I especially enjoyed the office hours I had with the many undergraduate students which I had the pleasure of helping. This research was made possible by the support of the United States Air Force grant (AFOSR #FA9550-17-1-0028) and later on in my research by the National Science Foundation grant (NSF #1646367).

To my coworkers who saw me through the final stretch of my studies - I thank Alex Black for acting as my therapist on many occasions whenever I became overwhelmed and needed a distraction to hear about the newest computer equipment he ordered. Luc Robitaille always expressed a curiosity in my research which I very much appreciated. I give a special thanks to Alexi Moutafis for being many things to me: a coach, a mentor, and a great friend through professional, personal, and academic challenges alike. To the entirety of TMC-AMETEK, colleagues past and present, thank you for your patience, support, and cooperation with me while I pursued higher education.

To the friends I made during my time at WPI, I thank you for making even the longest and most tedious days enjoyable. Brendan Sullivan taught me about what it truly takes to be dedicated about something you are passionate about through the dedication he often displayed in finishing entire bags of chocolate without hesitation or complaint. To the FBC, thank you for your patience while I finished this body of work. Now, it is time to ready up!

Much of the work for this dissertation took place remotely during the COVID-19 pandemic. I thank my in-laws Francisco and Gabriela Marin for the immense support and generosity of the room-and-board during this period of time, as well as everything they have done and continue to do for me. I thank Fran for being an amazing role model and hope I can one day live up to the example he has set. I was lucky to know one of the sweetest pups, Miss Holly Marin, who always managed to convince me to give her half of my lunch meat. We miss her dearly, and I know her presence and support during long weekends and nights of work helped make this dissertation possible. I thank Gus for introducing me to Rubens and for being a great entrepreneurial role model.

To my parents, Doreen and Leo, I want to thank you for your unrelenting support growing up and for always helping enable me with whatever I was passionate about. The experiences you helped provide for me growing up shaped me into who I have become today and I recognize how fortunate I was to have two parents who care so much. I also thank my furry companion, Miss Ruby, for her camaraderie through late night slumber parties while I finished this dissertation.

Last, but not least, I thank my wife Natalie for her patience and support through the trials and tribulations that came with my pursuit of a PhD. If it was not for her I am not sure I would have made it through this experience. I truly could not ask for a better partner in life. I believe she has earned an honorary degree just for all of the time and things she has had to sacrifice in order for me to accomplish this goal.

Abstract

This dissertation develops a method for coupling sensor configuration and path-planning, two tasks often decoupled, as a means to reduce an agent's exposure to unknown but observable threats. This method exploits *task-driven* sensor configuration for gathering information with statistical relevance to the path-plan, the *region of interest*. The method addresses a sensor's field of view and the effect of overlapping field of views. Additionally, the sensor configuration optimizes the trade-off between sensor quality and quantity of information. We show that by coupling sensor configuration with path-planning in this manner, a near-optimal path-plan for the agent can be discovered with fewer observations than information-greedy approaches.

Several scenario-specific variations are devised, such as *greedy batch* and *exploration efficient* modifications which handle cases where the path-plan optimality requirement is strict and when there are many sensors available, respectively. Extensions to multi-agent multi-goal scenarios are provided. We address computational performance through qualitative sensor configuration, which is achieved with cluster analysis, yielding suboptimal fast-approximations to sensor configuration for path-planning. These topics are decorated with numerical studies and examples displaying their benefits and performance guarantees.

Building upon the prior topics, a self-adaptive surrogate optimization function was devised which enables sequential sensor configuration with near-optimality guarantees. The adaptive component balances the *exploration-exploitation* trade-off in discovering the optimal path-plan. We utilize the sequential sensor configuration for scenarios such as *heterogeneous* sensor configuration whereby the threat environment is *multimodal*. We consider scenarios in which there may be a penalty for *waiting* for an optimal-path plan and develop an *active* sensor configuration strategy which can be applied to *dynamic* and evolving spatiotemporal threat environments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Problem Statement

Parallel advances in computer hardware and autonomy have continuously pushed the ceiling of what is possible in society. Cars can now drive themselves on highways, robotic vacuum cleaners can now clean floors on their own, and our phones are intelligent enough to give us alerts and news we care about without our asking. Autonomy reduces the burden of human effort for tasks that are considered to be dirty, dangerous, or demanding. In this dissertation, autonomy is attained in three ways: (1) autonomy of a primary acting agent, (2) autonomy of a network of sensors, and (3) autonomy of data fusion and comprehension.

Path-planning and motion-planning are significant benefactors of autonomy. Path-planning is the field of study for finding a set of waypoints for an agent, such as a mobile robot, to follow. Motion planning is a field of study for finding a trajectory based on the kinematic and dynamic characteristics of the autonomous agent (turning radius, ability to accelerate or decelerate, etc.). Both path and motion planning problems often consider finding paths or trajectories to minimize exposure to a *threat*, time to complete tasks, and obstacle avoidance within an environment. The metric for determining the optimality of such a path is known as “path cost”, which is a particular metric that a path planning or motion planning algorithm attempts to minimize.

The internet of things, or IoT for short, has enhanced the scope of intercommunication amongst devices. Networks of devices can connect locally and across the Internet to form a web of connections for data transmission and cohesion. Smart homes are a great example of such a network, where we now have sensors and actuators around our homes that can control local interior climates, secure our homes, and alert our phones when undesirable events occur. Surveillance activity performed by unmanned aerial vehicles (UAVs) typically involves setting target locations for the UAVs to collect data. This is known as *sensor placement*, and it typically involves using metrics from past observations and the environment to help inform the best sensor placement for data collection. Using this concept of IoT, a network of sensors may be distributed amongst UAVs, and after determining the optimal sensor placements, they may be deployed to those positions. The utilization of a sensor network can significantly reduce data collection times since a single sensor unit would not need to reconfigure itself to each optimal position for data collection, and the tasks may be delegated across the network. The goal of this dissertation is to investigate how to best place sensors within an unknown environment to best assist in collecting data pertinent to the path planning efforts for the acting agent.

Data returned from sensors are not always homogeneous, and often a task at hand may require a *heterogeneous* mixture of sensors to adequately observe the environment, also known as *multi-modal sensing*. For example, UAVs may be equipped with various sensor types: electro-optical, infrared, lidar, etc. The data obtained from these sensors are not the same, as electro-optical may return a 2D collection of pixels with RGB values, infrared can return a 2D collection of pixels with thermal intensity, and lidar is often a 3D point cloud representing relative distance. The act of combining and unifying the multimodal data is called *sensor fusion*. However, determining the significance of data typically requires human input and monitoring. Many methods for combining multimodal data require human intervention to provide the context-based correlations and statistical significance of the data that is received and may be received in the future. Multimodal data fusion typically has the requirement of at least one human expert and often lacks any streamlined automatic approach for self-determination of contextual significance of the data by a machine. By developing a method that can separate, or at least minimize, the need for human input to the context-based significance of sensor data, its statistical significance, and the unknown correlations between modalities, we may achieve autonomy of the data fusion process of multimodal data.

The synthesis of the planning, sensing and data fusion problem yields an integrated sensing and planning scheme. By coupling the sensor configuration and planning problem, the sensor configuration can be geared towards optimally measuring the region of interest of our planning objective. Rather than having to sense an entire environment, by utilizing uncertainty metrics pertaining to the planning objective we may focus sensing efforts in a *task-driven* manner. The novelty of this work is that the sensor reconfiguration and multimodal data fusion is performed in context to a planning problem, thereby minimizing the iterations required to solve the planning problem and at the same time minimizing the need for human input and interaction.

1.2 Background and Literature Review

Research on sensor placement is mainly independent from path-planning and motion-planning research, and as a result the methods produced are fairly disjointed. Sensor node positioning problems are typically classified into static or dynamic positioning. Static approaches are aimed at achieving optimal situational coverage without future repositioning. Dynamic sensor node positioning involves updating the positions to improve coverage metrics as environments or situational changes dictate (Younis and Akkaya, 2008). Coverage techniques are typically separated into three main types: topology-based coverage, ratio-based coverage, and object-based coverage. Some specific methods involve grid-based deployment, Delayuny Triangulator, and Voroni-Diagrams (Farsi et al., 2019). Coverage models include disk-based models, probabilistic coverage models, information coverage models, and confident information coverage models. These models are used in combination with various techniques to optimize deployment costs, network lifetime maximization, and coverage hole detection and healing. In heterogeneous sensor networks, data fusion efforts focus on various coverage optimization techniques that utilize probabilistic methods for sensor density optimization, heterogeneous information coverage and sensor scheduling (Deng et al., 2018). Lastly, various advances in machine learning have impacted the sensor network field, such as improving coverage, data aggregation, and anomaly detection tasks. Reinforcement learning has enabled a two-staged sleep scheduler that maximizes network lifetime while maintaining proper sensor coverage. Multi-sensor heterogeneous data fusion has improved accuracy when utilizing a distributed Bayesian framework across the network. Anomaly detection and removal in sensor networks has been improved through the combination of SVM and DBSCAN techniques. Adversarial attacks can be protected against through the use of a decision tree based scheme (Praveen Kumar et al., 2019).

Both path and motion planning are rigorously studied and typically involve finding obstacle devoid, cost minimizing, or utility maximizing paths. Control architectures in path and motion planning operate under a think, then act scheme (Anavatti et al., 2015). Path planning techniques are typically divided into classical and heuristic based approaches. Popular classical approaches are cell decomposition based on 2d wavelet transforms, roadmap approaches such as Voronoi diagrams and Rapidly Exploring Random Trees, and artificial potential fields. Heuristic approaches involve wavelet decomposition, A* and its variants, artificial neural networks, fuzzy logic, and genetic algorithms (Cowlagi, 2014; Injarapu and Gawre, 2017; LaValle, 2011; Mac et al., 2016; Patle et al., 2019). Common challenges in path planning involve minimizing path length, determining optimality conditions, completeness, cost-efficiency, time-efficiency, robustness, and collision avoidance (Aggarwal and Kumar, 2020).

Ontology-based Knowledge Representation techniques can combine domain and context-relevant knowledge such as temporal, spatial, and semantic knowledge. This enables high level planning using symbolic classical planning, constraint-based time-flexible planning, and domain-specific representations (R. and Uma, 2018).

Some path planning methods explicitly are built to solve path planning in time sensitive situations. Time Enhanced A* (TEA*) is an incremental algorithm that accounts for other robotic agents in an effort to avoid collisions and deadlock situations by utilizing time dependent state checks. Time Windows method checks a set of candidate paths for feasibility and also checks for conflict through overlapping windows. It enables dynamic routing and can be used in applications such as pickup and delivery, logistics routing, and multi-AGV dynamic routing problems (Costa and Silva, 2019). Velocity estimation can aid in predicting motion of dynamic objects, and hidden Markov stochastic models, Kalman filtering, and scene-flow estimations can provide such estimations (Anavatti et al., 2015).

Sampling-based motion-planning methods have allowed real-time kinodynamic, real-time, and optimal planning. Sampling algorithms include RRT and Probabilistic Road maps. In the sampling operation, various sampling strategies and metrics are used to extend, guide sampling exploration, and prevent collisions. These techniques can solve high dimensional problems in short amounts of time. Optimal sampling methods such as RRT* extend performance to satisfy optimality guarantees (Elbanhawi and Simic, 2014).

Various path planning methods have been inspired by naturally occurring biological phenomena. Swarm intelligence methods include particle swarm optimization and ant colony optimization. There are also evolutionary algorithms such as genetic algorithms and differential evolution. Neurodynamic models such as artificial neural networks and variants such as the Hopfield network have also been developed for path planning (Li et al., 2019).

Autonomous vehicles, in general, operate under a decision-making hierarchy that involves route planning, behavioral decision making, motion planning and vehicle control. Specifically, motion planning is dependent on the behavioral decision making within the current situational context. The motion planning effort is then required to determine a trajectory that is dynamically possible by the vehicle model. The behavioral decision making is dependent on the route planning, or path planning output. Thus, we see that the relationship between path planning and motion planning is a dependent and often sequential process. Motion planning is largely a context-based problem where the planning operation may need to consider both static and dynamic obstacles and even utilities of performing certain motions. Path planning is a necessary precursor as it determines a feasible and sometimes optimal generalized path. Trajectory planning suits dynamic environment planning where kinematic constraints are taken into account. Variational methods, such as psuedospectral approximation schemes like Pontryagin's minimum principle, are often used to address nonlinear continuous optimization trajectory planning problems. Continuous geometric techniques involve using Bezier curves, Spline curves, Polynomial spirals, arcs, and clothoids. Model predictive control solves trajectory planning as a nonlinear optimization problem and provides a sequence of control input that is both predictive and constraint-aware. Autonomous vehicle motion planning problems also consider graph based abstractions of motion planning problems. These include lane graphs, geometric methods such as Voronoi diagrams, sampling-based methods such as probabilistic roadmaps, and graph search strategies. Incremental optimal motion planning attempts to incrementally plan based on dynamic motion reachability (Paden et al., 2016; Sharma et al., 2019).

In the context of optimal motion planning, various constraints and decision variable representations are taken into account. Objective function formulations typically look to find minimum length and improve path smoothness, and decision variables can be anything from joint positions to state cells and trajectory segments. Constraints come in the form of collision avoidance, pose constraints, and dynamic constraints such as non-holonomic constraints (Yang et al., 2019).

Path-planning and motion-planning in 3D spaces, typically for UAVs, involves many of the same path planning techniques in 2D spaces, but with some caveats and extra considerations. UAV path planning, for example, often takes into account stealth, physical feasibility, mission performance and real-time implementation details (Zhao et al., 2018). Modifications to heuristic algorithms sometimes involve dimension reduction techniques such as the creation of virtual terrains prior to performing A*. Artificial potential fields are sometimes modified by adding guiding poles to avoid local minima, and sometimes virtual environments are created. Path smoothing is occasionally applied post path plan so as to create more fluid trajectories for UAV dynamics (Quan et al., 2020; Song et al., 2019). Typical UAV applications are scene reconstruction, environment exploration with obstacle avoidance, and even aerial cinematography (Zhou et al., 2020).

In the following sections, more specific topics are reviewed as they relate directly to the proposed work. First, field regression techniques are evaluated for generating representations of the environment to be modeled. Then, specific path planning and motion planning algorithms addressing planning under uncertainty are detailed. Optimal and near-optimal sensor placements will then be discussed to provide an outline of existing techniques for placing sensors to model environment dynamics. This leads to an evaluation of multimodal sensing methods, and how we may fuse the data aggregated from sensor data. These topics are all then combined into a discussion of current research in interactive planning and sensing methods in time-varying environments.

1.2.1 Model Estimation, Inference, and Forecasting

Estimating an environmental model requires obtaining data that can be utilized by a regression technique for fitting that data, or rather conditioning on the input. Not all regression methods are created equal, and various approaches have model and context dependent situations in which they perform their best. Numerical studies on various regression models within regression families has been conducted to study these trade-offs. Regression families such as Bayesian models, Gaussian Processes, Neural Networks, and Support Vector Regression, to name a few, were numerically evaluated and ranked in different situations using Root Mean Squared Error (RMSE) and Mean

Absolute Error (MAE) metrics (Fernández-Delgado et al., 2019). Most regression models, however, all fall under one unified representation known as a mixture of linear models. Linear models such as weighted linear least squares and weighted sum of basis function models such as radial basis function networks, all can be thought of a specialized approaches to mixtures of linear models. The main differences, whether they are based on weighted sum of basis functions or not, is in the number, position, and size of the basis functions, and their respective hyperparameters (Stulp and Sigaud, 2015).

Some regression techniques utilize Bayesian methods, which allow for uncertainty, the variance and confidence interval, to be computed alongside the estimate. Bayesian inference is a classical method in statistics for updating the hypothesis (prior) as more information or data is presented. Bayesian regression is a parametric approach that utilizes an informative prior combined with the likelihood to determine the posterior mean and uncertainty distributions. Such approaches can be performed on typical ridge regression, logistic regression, and linear regression formulations (Andrew et al., 2014). Another such model is Gaussian Mixture Regression, which represents the output as a set of Gaussians and is fit using expectation maximization (Calinon et al., 2007). The technique known as Gaussian Process Regression (GPR) is a nonparametric (infinitely parametric) method for estimating the mean and uncertainty of a model. In this method, a covariance function is represented by a kernel and is typically assumed to have a mean centered about 0. The GPR method is a special case of Radial Basis Function Networks with each data point being the center of a basis function and each basis function has a width determined by the kernel, also known as the Gram matrix (Rasmussen and Williams, 2006).

Due to the nonparametric nature of GPR and its ability to additionally output uncertainty since it is a Bayesian approach, makes it very desirable for many applications such as the one that will be described later on. Some potentially limiting factors of the GPR method is that it requires the selection of an appropriate kernel (for the covariance function) as well as the determination of hyperparameters for these kernels. In addition, it has a time complexity of $\mathcal{O}(n^3)$, which can be difficult to handle unless using high end computers when dealing with large data sets. Methods for dealing with this include Reduced-rank approximations of the gram matrix, the Nystrom Method, Bayesian Committee Machines, and the Informative Vector Machine method (Fine; Herbrich et al., 2003; Tresp, 2000; Williams and Seeger, 2001).

The determination of the composition of the kernel is an important requirement for proper structural modeling. One approach, the compositional kernel search method, scores individual families of kernels, and then inspects the products of other various kernels along with additions of other kernels recursively. Thus, it attempts to discover the best structure by first finding the best kernel family and then combining with more fine tuned kernels (Duvenaud et al.). The Approximate Bayesian Computation and Sequential Monte Carlo method provides a Bayesian approach to finding optimal kernels, mean functions, and their hyperparameters while additionally providing confidence intervals for the kernel selections (Abdessalem et al., 2017). Methods focusing on determining the optimal hyperparameters alone are studied using Rprop, which is a gradient-based optimization technique that uses adaptive update steps (Blum and Riedmiller, 2013). Through the use of Bayesian Regularization in the model selection phase, the effect of over fitting is reduced by removing additional regularization parameters analytically with the reference prior. It has been shown to outperform the expectation-propogation based Gaussian Process in various cases (Cawley and Talbot).

Gaussian processes typically assume i.i.d. noise at every sample, but there are some methods that can account for non perfect distributions. For data with a skew normal distribution, the Gaussian Process Regression with Skewed Errors (GPRSE) method is a proven technique to handle such a distribution (Alodat and Shakhatreh, 2020). Alternatively, an intermediate combined support vector regression (SVR) and extreme learning machine (ELM) method can be used to infer the noise variance prior to using weighted GPR to infer the model estimates (Li et al., 2020). Besides odd noise distributions, methods for handling missing data have also been studied. The Expected Gaussian Kernel assumes the distances of missing data are Gamma distributed and it attempts to determine the expected value of the kernel function between two pairs of argument vectors with potentially missing data (Mesquita et al., 2019).

Deep Gaussian Processes are deep belief networks that combine the benefits of deep network architectures with the benefits of Gaussian Processes, namely the ability to characterize uncertainty in estimates. They also have the ability to determine better estimates of uncertainty and better fits overall. Using the approximate variational marginalization method DGPs can have automatic structure design since it marginalizes away the latent space (Damianou and Lawrence, 2013). Another way to do this is through a moment-based approximation that utilizes a GP to approximate (Lu and Shafto, 2020). Scalable deep kernels have been explored using a base spectral mixture kernel combined with a deep architecture for automatic deep kernel learning (Wilson et al., 2015).

Forecasting subsequent state estimates in discrete or continuous time-varying situations requires further consideration of how state estimates and their uncertainties are propagated through time and how they can be used to inform future estimates. Gaussian processes alone can be used to model mean and uncertainty estimates, where the uncertainty drops whenever an observation is made, and grows as time passes (Roberts et al., 2013). A Stochastic Recursive Gradient Propagation approach to training sparse GPs, which utilizes inducing point approximation for computational efficiency combined with Bayesian recursive estimation, enables an update feature comparable to a Kalman filter for online learning and sequential hyperparameter estimation (Schürch et al., 2020). Combinations between GPs and Kalman Filters have been studied to fuse the benefits of both for applications requiring spatial-temporal field estimators (Reece and Roberts, 2010). GPs have also been combined with Unscented Kalman Filters for nonparametric state estimates using stochastic approximation, also known as the unscented transform (Ko et al., 2007). A variant of the Recurrent Neural Network structure, the Long Short-Term Memory (LSTM) cell, has been utilized for forecasting multimodal and can be supplemented with physical state evolution models (Chung et al., 2019; Greff et al., 2017). Combinations of LSTM cells with GP inputs have been developed so that uncertainty estimates can influence LSTM model predictions appropriately and provide the uncertainty at the posterior of the LSTM cell (Lam et al., 2019). Ensemble learning using Random Forests, GPRs, and LSTM outputs have shown success in better predictive ability, especially if forecasting is required in multimodal scenarios (Wang et al., 2020).

1.2.2 Path-Planning and Motion-Planning under Uncertainty

Path planning and motion planning were previously briefly discussed in the previous section, however, we focus herein on distinctive methods when there is uncertainty in the planning task. Uncertainty in path and motion planning can arise from dynamic environments, modeling errors, and measurement noise. Robot motion planning in dynamic environments is a significant research area that addresses uncertainty that arises from the environment specifically (Mohanan and Salgoankar, 2018). Motion planning techniques are vast, and a notable method is the concept of configuration spaces where the objective is to find a continuous sequence of free spaces. Velocity based methods include partial motion planning, which determines the best partial motion given time constraints and uses a tree created with probabilistic techniques, and predictive temporal motion planning (Thorn).

Dynamic environments with high levels of uncertainty typically need to utilize probability based motion planning techniques to adequately model the uncertainty. These are typically solved utilizing dynamic programming techniques (Mohajerin Esfahani et al., 2016), or even using the unscented transformation for probabilistic spread (Sun et al., 2014). Rapidly exploring Random Trees (RRTs) are often used to generate optimal and valid paths. This has been combined with sampling based methods and the linear quadratic regulator for path stability in uncertain environments (Li et al., 2017). They have also been used in conjunction with growth stagnation and re-grow techniques for the RRTs to solve planning problems with many blind regions (Gu et al., 2013). The Convergent RRT, Convergent Multiple-Restart RRT, and Convergent Stable Sparse RRT methods are asymptotically near-optimal solutions for kinodynamic planning with uncertain initial states (Liu et al., 2019).

Both Markov Decision Processes (MDP) and Partially Observable Markov Decision Processes (POMDP) are stochastic motion planning techniques that are used to solve uncertainty in the action and state of motion planning problems. They can be combined with high level linear temporal logic task formulas to find optimal policies for system trajectory (Guo and Zavlanos, 2018). An online POMDP has been developed to handle dynamic uncertain environments by combining POMDP framework with point-based planning to find the approximate value function for the online planning components (Ji and Li, 2015). Guided Cluster Sampling (GCS) is a point-based POMDP planner that handles continuous states and actions. It is proven to converge to the optimal policy in sufficient time (Kurniawati et al.).

Simultaneous Localization and Mapping (SLAM) algorithms attempt to reduce localization and environment uncertainty in parallel, and they can also incorporate some planning element (SPLAM) (Chakravorty and Saha, 2008; Toriz Palacios and Sánchez López, 2017). SLAM has been combined with both Extended Kalman Filters and Unscented Kalman Filters to improve agent state estimates and estimates of obstacles and other threats in an environment (Gharib and Esmaili, 2019; Tao et al., 2007). Rapidly exploring Randomized Trees have been combined with SLAM to perform motion planning with localization, sensing, and mapping uncertainty by adding an extra dimension of uncertainty to the C-space (Huang and Gupta, 2008). SLAM methods primarily operate with interoceptive sensors (i.e. sensors local to an agent or agents), during active localization and mapping of an uncertain environment. The work presented in this dissertation differs in that we consider exteroceptive (i.e. decoupled from the agent(s) themselves) which obtain information for the agent. Critically, the procedures in this body of work act as a high-level mission planner for the agent whereas in SLAM approaches the agent is the information gatherer and self-planner.

Reinforcement learning has also been used to handle uncertainty in motion planning problems. It has been used for calibrating objective function weight parameters with a probabilistic policy and Boltzmann distribution for likely actions (Igarashi, 2001). Applications of reinforcement learning in motion planning are shown to solve battery optimization, under-actuated motion planning and control of underwater vehicles using reinforcement learning based on Markov Decision Processes, and has been used as global path planners for obstacle avoidance (Kawano, 2005; Sichkar, 2019; Soni et al., 2019). Furthermore, research into deep reinforcement learning, in an actor-critic architecture, for path planning in unknown environments with continuous action spaces allows for motion control parameter discovery (Yan et al., 2018). Within the machine learning domain, the Runtime Stochastic Ensemble Simulation technique enables online planning and utilizes Monte Carlo simulations for collision likelihood, and combines with RRTs to generate best paths (Chiang et al., 2016).

1.2.3 Optimal and Near-Optimal Sensor Placement

Expanding upon the sensor placement strategies previously discussed, the task of sensor placement as it relates to spatiotemporal environments and their uncertainties in an optimal or near-optimal manner is non-trivial. Such optimal sensor placement has been studied for estimation of threats such as the dispersal of gases, specifically for volcanic ash, and has been accomplished using distributed network techniques (Demetriou and Uciński, 2011; Madankan et al., 2014; Uciński, 2012). Metrics of performance to characterize sensor placement optimality are entropy minimization, Kullback-Lieber divergence, Hellinger distance, mutual information, and maximization of various properties of the Fischer Information Matrix (Cochran and Hero, 2013).

The objective of the sensor placement varies based on the context-relevant nature of the problem at hand. Placement can be based on uncertainty minimization, spatial coverage, sensor network lifetime, and even placement for optimal communication between the sensors themselves (Ramsden). An additional consideration is locational error of the sensor placements. An adaptive placement technique that accommodates uncertainty in sensor location that utilizes a Gaussian Process framework for environmental modeling has been developed as one potential way to remedy this issue (Nguyen et al., 2017). Optimal sensor placement for spatial phenomena modeled by Gaussian Processes has gained attention. Methods for finding sensor observation paths, and individual placements for Gaussian Process based estimation such as the PSPIEL algorithm, which is both robust against the environment and can be solved in polynomial time (Krause et al., 2010; Van Nguyen et al., 2013). Extension placement methods in Gaussian Process derived environment models for heterogeneous sensor types has been explored, such as greedy algorithms for fully equipped sensor payloads, or sensor modality deficient payloads (Sun et al., 2018).

Decentralized methods for optimal sensor placement for target tracking have seen success utilizing the determinant Fischer Information Matrix alongside an extended Kalman Filter for tracking (Martínez and Bullo, 2006). The Rapid Random exploring tree with Linear Reduction method enables sensor placement with the goal of minimizing the required number of placements without information loss in graph based models (Chen et al., 2018). Optimal sensor placement in spatially networked processes with contact dynamics has been studied using a combination of sensor influence waves, probabilistic target models, spatial weight maps, and a genetic algorithm for iterative optimization (Krishnamurthy and Khorrami, 2018).

Active perception with multi-robot systems has been studied in literature, namely for target detection and tracking applications. Problems in this domain are classified as target detection when no target has been identified, where perception is either performed with static surveillance or mobile search which can be performed with capture, probabilistic search, or hunting methodologies. Target tracking is distinguished as a target localization problem when multiple viewpoints can be considered or a monitoring problem when a single viewpoint is provided (Robin and Lacroix, 2016). The four major control techniques for moving target tracking are cooperative tracking, cooperative multi-robot observation of multiple moving targets, cooperative search, acquisition, and track, and multi-robot pursuit evasion (Khan et al., 2016). Fair multi-target tracking is presented in (Banfi et al., 2015), which attempts to provide unbiased cooperative monitoring of targets. Distributed action and target assignment for multi-robot multi-target tracking is presented in (Sung et al., 2018), which maintains communication constraints and accounts for limited sensor field of view all while providing tracking performance guarantees. In (Verma and Ranga, 2020), the authors describe methods for multi-robot coordination when each robot is confined to disjoint regions of space. In this dissertation, we utilize a sensor network which can be a multi-robot system that identifies information theoretical coverage and multi-robot exploration for an agent's path-plan rather than continuous tracking.

The sensor network we consider can be realized as a camera network using exteroceptive electro-optical imaging sensors. Modeling coverage in camera networks literature addresses coverage criteria (field of view, resolution, and focus) as well as common problems such as static and dynamic occlusion (Mavrinac and Chen, 2013). In contrast to fixed macroscopic camera imaging for industrial control applications, which have incomplete visibility, the authors (Kamezaki et al., 2016) detail an adaptive imaging system which performs target following, camera posture control, and adaptive zoom control for a human-in-loop visibility for precise operations. Utilizing fixed cameras, they selectively assign roles and coordinate their field of views to best assist an operator. The work we present focuses on not only field of view coverage, but also information quality. We aim to configure sensors to locations which provide maximal information coverage to path-planning tasks.

1.2.4 Multimodal Sensing and Data Fusion

The previously mentioned sensor placement techniques mainly operated on an assumption of unimodal/homogeneous sensors. Multimodal sensing and data fusion deals with heterogeneous data and context-relevant basis in which they can be fused. Information fusion has 7 distinct levels: 1) Level 0 - Sub-Object Data Assessment, 2) Level 1 - Object Assessment, 3) Level 2 -

Situation Assessment, 4) Level 3 - Impact Assessment, 5) Level 4 - Process Refinement, 6) Level 5 - User Refinement, and 7) Level 6 - Mission Management (Blasch et al., 2019; Llinas et al., 2016; Snidaro et al., 2016). Levels 3 and 4 are the two levels that most directly apply to optimal sensor placement and the resulting path planning efforts as a result of the fusion efforts. High-level information fusion system design is explored, with topics including situational awareness, context-based adversaries, and context-based network modeling (Blasch et al., 2012, 2013). Similarly, low-level information fusion has been investigated, namely in target tracking and identification, using a feature fusion model for object identification across modalities and also through use of stochastic integration filters (Dunik et al., 2015; Kahler and Blasch, 2010).

Knowledge of the domain is critical, and context-based relevance of data is critical to properly fusing correlated and uncorrelated data. There are many assumptions that may be made in regards to context-based threat assessment systems. These might include: context-based data can be acquired to fill knowledge gaps, data collection is unbiased, target signatures remain constant, data quality is measurable, the data itself has the ability to be fused, the fusion strategy itself is fixed, and errors are quantifiable to name a few. Successful threat assessment systems which require some level of information fusion depend on the ability to validate context-based assumptions (Israel and Blasch, 2016).

Deep learning techniques for multimodal representation have three main frameworks: joint representation, coordinated representation, and encoder-decoder. Deep learning models for multimodal representation include probabilistic graphical models, multimodal autoencoders, generative adversarial networks, and deep canonical correlation analysis (Guo et al., 2019). Bimodal deep autoencoders and sparse restricted Boltzmann machines have been used in practice for cross modality learning in audio and video applications (Ngiam et al.). The measure of effectiveness of deep learning methods on such tasks is based primarily on the quality, robustness and overall information gain from utilizing a particular model (Blasch et al., 2018).

Some specific methods for optimal sensor placement, path planning, and field construction with relevance to multimodal sensory data have been previously mentioned in earlier sections. An upstream fusion technique that synthesises deep learning, topological analysis over probability measure, and hierarchical Bayesian non-parametric recognition models is capable of automatically finding inter-modality dependencies, hierarchical representation discovery, and the removal for the necessity to specify or encode features a priori (Garagić et al., 2018). This is powerful because one of the largest challenges in multimodal fusion is the context-based relevance and identification of feature significance and correlations. The use of joint manifold learning fusion for heterogeneous upstream data fusion enables discovery of embedded low dimension representations for target tracking (Shen et al., 2018). Non-linear data fusion methods such as the particle filter, sum-of-gaussians, and likelihood estimation methods have been shown to more aptly solve multimodal data fusion tasks in comparison to techniques such as the extended Kalman Filter (Durrant-Whyte and Henderson, 2016). Gaussian process based methods have also seen much success in the way of multimodal data fusion, and have been applied to target tracking applications and have been shown to outperform extended and unscented Kalman Filter methods (Xie et al., 2018). Heteroscedastic GP, multi-output and multi-dependent GP models have shown success in fusing multimodal sensory data (Vasudevan, 2012). A heteroscedastic and non-stationary GP has been used for multimodal sensor fusion for applications such as vehicle single track models for lane change maneuvers (Rhode, 2020).

1.2.5 Interactive Planning and Sensing in Time-Varying Environments

The motion planning and sensing tasks previously described are traditionally disjoint problems solved independently of one another. However, research into methods leveraging a combined interactive planning and sensing have been recently gaining interest. Initial research interests included gimbaled electro-optical and infrared sensors on a UAV platform utilizing an information-driven metric for sensor placement with limited field of view within the environment to generate a concurrent path plan (Skoglar et al., 2006). A Parallel Hierarchical Replanner scheme was devised to develop dynamic, reactive path plans given multi-agent data fusion of the environment, which lends itself to a parallel sense and plan scheme (Allen et al., 2009). Lastly, Markov Decision Processes have been used to generate path plans by harnessing uncertainty in time-varying environmental parameters in an effort to minimize planning uncertainty and exploit environmental benefits, similar to the effort of utilizing observational data of the environment and the statistical properties to jointly plan the optimal path (Al-Sabban et al., 2013).

The idea of an Interactive Planning and Sensing (IPAS) framework was theoretically formalized using a task-driven sensor placement scheme. Task-driven is in contrast to typical information-driven metrics which attempt minimize entropy in an entire environment, since task-driven focuses on a subset of the environment of direct relevance to the planning task (Thrun, 2002). Using a set of parameters with support over a compact workspace, sensor placements were selected that best minimized the uncertainty along and within a region near the current optimal path. The method is proven to be near-optimal with a high probability and is guaranteed to converge. In addition, it also attempts to minimize the number of iterations to converge and has been shown to converge faster than methods such as frame potential (Cooper and Cowlagi, 2019c). An incremental 3D path repair method was combined with the IPAS scheme for a UAV in partially known environments that allows for 3D shortcut detection and a policy for a sensor's Field of View to best serve the IPAS framework (Du and Cowlagi, 2017).

The IPAS methodology has been similarly applied to spatiotemporally evolving environments where sensor scheduling to observe task-driven information is performed over a finite time horizon, and theoretical conditions in which it converges have been devised (Cooper and Cowlagi, 2019b). In that study, the process was broken up into measurement collection, planning, and sensor reconfiguration time steps that require future research study for convergence analysis. While this method and the previous IPAS method were centralized models, two decentralized counterparts were developed that enable distributed estimation, both using a distributed information-consensus filter to update local estimates and covariance. The two methods featured a ”Deployable-recon” which allows sets of sensors to be disjoint in terms of communication links, and a second was ”Actor-connected” meaning that all sensors had to be within range of communication with one another as well as the primary actor (Cooper and Cowlagi, 2019a).

1.3 Dissertation Overview and Statement of Contributions

The primary goal of this dissertation is to provide a framework for sensor configuration that provides maximal information gain to the path-planning task in minimal iterations while minimizing an actor's threat exposure.

1.3.1 Dissertation Overview

After thorough literature review, an apparent technological gap is noticeable between algorithms which attempt to optimize sensor placement and algorithms that are concerned with path-planning. These tasks are often decoupled, with few exceptions. While information-greedy sensor placement algorithms may exist, they do not necessarily exist in a way that is optimal for the path-planning decision making effort. In this dissertation, we aim to create a robust framework that directly couples sensor configuration and path-planning.

One of the primary items of direct concern is understanding the trade-off of sensor quality versus quantity when performing sensor configuration. This dissertation directly addresses sensor field of view and the effect of correlated and overlapping sensor measurement regions as it pertains to the path-planning effort. Additionally, we explore various application specific sensor configuration algorithmic variations to best serve specific use cases for path planning such as safety critical scenarios and situations when a large quantity of sensor are configurable.

Additionally, this dissertation addresses techniques which exploit the proposed framework with fast approximation sensor configuration techniques and surrogate optimization functions for enabling sequential sensor configuration optimization.

In a macro perspective, we aim to develop a sensor configuration framework which enables heterogeneous sensor configuration in an information theoretical framework with respect to the path-planning. Of importance in this framework is the ability to operate in active planning scenarios and dynamic environments. We address the computational complexities of the methods developed and discuss the trade-offs when applying sensor configuration approaches.

1.3.2 Statement of Contributions

The following are a select list of summaries of the contributions made by this dissertation toward a coupled sensor configuration and path-planning framework that is capable of managing heterogeneous sensor networks for one or many agents and in static or dynamic environments.

Coupled Sensor Configuration and Path-Planning Minimizes Sensor Observations with respect to Information Greedy Approaches and Requires Fewer Sensors We develop an algorithm which couples sensor configuration to regions of interest with direct consequence to path-planning. The direct implementation utilizes a task-driven information gain metric that aims to quantify the information gained from observing path-plan relevant regions. We numerically show that the task-driven sensor configuration can complete the path-planning optimization with far fewer observations and sensors than information greedy approaches.

Coupled Sensor Configuration and Path-Planning Optimizes Path-Planning in Unknown Threat Environments We demonstrate that the coupled sensor configuration and path-planning algorithm converges to an optimality criteria that represents our confidence in path-plan optimality. By provably converging below this criteria, we show that the resulting path-plan is optimal within a statistical optimality guarantee.

Optimal Path-Planning for Multi-Agent Scenarios in Unknown Threat Environments We demonstrate the ability for the coupled sensor configuration and path-planning algorithm to scale to find many near optimal path-plans for multiple independent agents.

Batched Sensor Configuration for Safety Critical Applications In scenarios in which an extremely strict optimality guarantee is required for convergence, it may require multiple duplicated samples of the near optimal path-plan estimate to obtain this guarantee. To accommodate a strict termination threshold, we developed a greedy batched coupled sensor configuration and path-planning variation which enables multiple configurations by the sensors each iteration. We show that this lowers the required iterations to convergence in such applications.

Observation Efficient Sensor Configuration for Large Sensor Counts The coupled sensor configuration and path-planning algorithm can achieve near optimal path-plan estimates in fewer iterations than information greedy approaches. However, as the number of available sensors increases, the performance gap between the approaches shrinks. To counter this, we developed a variation of the coupled sensor configuration and path-planning algorithm that configures sensors over many statistically feasible near optimal path-plans. We numerically show this approach outperforms information-greedy approaches even when a large number of sensors are available.

Qualitative Cluster Analysis Driven Sensor Configuration for lowering Time Complexity In static, yet time critical, threat environments it is imperative that the computations for sensor configuration are performed quickly, even if suboptimal with respect to the task-driven information gain. We utilize cluster analysis methods along with the task-driven information gain to find fast-approximation sensor configurations. We demonstrate the time-savings and closeness in performance of the method against direct optimization of the task-driven information gain metric.

Surrogate Sensor Configuration Function for Enabling Sequential Sensor Optimization We present a surrogate sensor configuration method which has provable near optimal guarantees when optimizing optimizing sensor configurations sequentially. This reduces the high-dimensional optimization problem significantly. Secondly, we utilize a self-adaptive technique within the surrogate method that balances exploration and exploitation of the path-planning efforts. We show that this method outperforms directly optimizing the task-driven information gain metric and information-greedy approaches. Additionally, we show that the adaptive coefficient outperforms any fixed value for said coefficient.

Coupled Sensor Configuration and Path-Planning with Multimodal Sensors Practical applications may require a sensor network comprised of heterogeneous sensor payloads. We developed a coupled sensor configuration and path-planning algorithm which generalizes to heterogeneous sensor types and multimodal threat environments. We show that the sequential optimization enables selective sensing when the sensor network has various combinations of sensor payloads.

Active Coupled Sensor Configuration and Path-Planning in No-Wait Scenarios We demonstrate the ability to discover path-plans in situations where the acting agent is actively updating it's position and planning. The method focuses sensor configuration on finding the route, but also avoiding spatially proximal threats. We consider the agent's ability to wait and numerically show that the active strategy we develop minimizes path-plan suboptimality.

Coupled Sensor Configuration and Path-Planning for Time-Varying Threat Environments There exist many scenarios in which the threat environment has a dynamic profile as time passes such as spreading forest fires. We detail necessary updates to extend the framework for the coupled sensor configuration and path-planning methodology to handle time dependent environment modeling and sensor configuration.

Chapter 2

Coupled Sensor Configuration and Path-Planning

Path-planning for an autonomous agent is typically performed subsequent to the observation and modeling of it's surrounding environment. When the agent is “blind” to it's surroundings, it relies on extroceptive sensors to map and realize the environment. The separation between the path-planning and sensor configuration efforts leads to unnecessary and excess exploration of the environment prior to deciding upon a best path. The goal of this work is to *couple* the sensor configuration and path-planning objectives to minimize unnecessary exploration of the environment. We devise a centralized framework which deploys extroceptive mobile sensors to observe regions of direct relevance to the agent's path-planning efforts. In what follows, we attempt to answer the question: *how do we optimally configure a mobile sensor network to find a near-optimal path-plan in a minimal number of iterations?*

2.1 Problem Overview

Throughout this dissertation we utilize the following notations. We denote by $\mathbb{R}$ and $\mathbb{N}$ the sets of real and natural numbers, respectively, and by $\{N\}$ the set $\{1, 2, \ldots, N\}$ for any $N \in \mathbb{N}$. For any $\boldsymbol{a} \in \mathbb{R}^N$, $\boldsymbol{a}[i]$ is the i^{th} element of $\boldsymbol{a}$ and $\text{diag}(\boldsymbol{a})$ denotes the $N \times N$ diagonal matrix with the elements of $\boldsymbol{a}$ on the principal diagonal. For any matrix $A \in \mathbb{R}^{M \times N}$, $A[i, j]$ is the element in the i^{th} row and j^{th} column. $\boldsymbol{I}_{(N)}$ denotes the identity matrix of size N.

The agent operates within a compact square planar region called the *workspace* $\mathcal{W} \subset \mathbb{R}^2$. Consider a uniformly-spaced square grid of points $i = 1, 2, \ldots, N_{\text{g}}$ and a graph $\mathcal{G} = (V, E)$ whose vertices $V = \{N_{\text{g}}\}$ are uniquely associated with these grid points. The set of edges E of this graph consist of pairs of geometrically adjacent grid points. In a minor abuse of notation, we label the vertices the same as grid points. We denote by $\boldsymbol{p}_i = (p_{ix}, p_{iy})$ the coordinates of the i^{th} grid

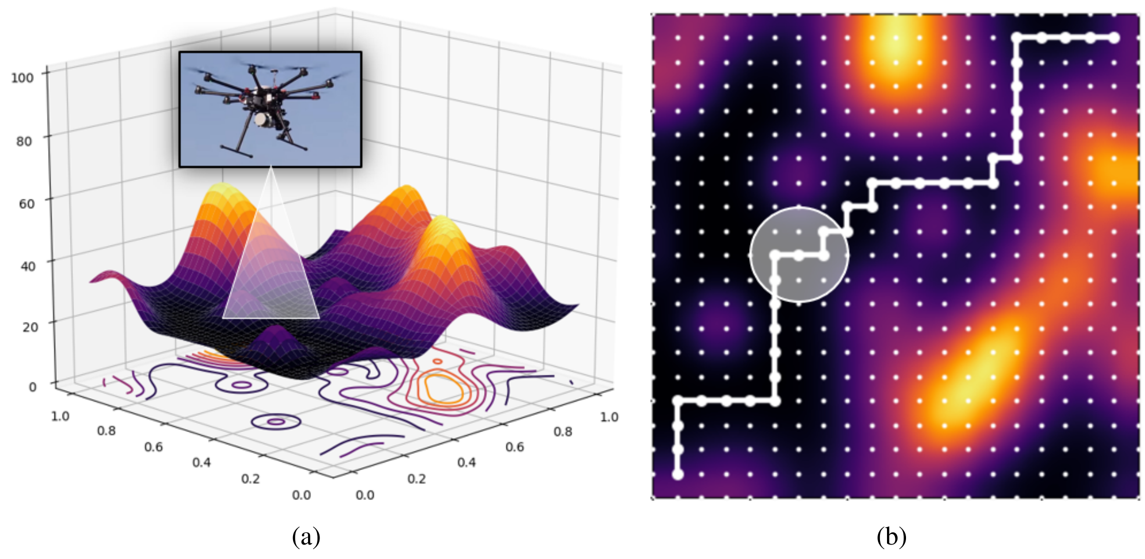


Figure 2.1: (a) Example threat field with a sensor observing it within a particualr FoV. (b) A direct realization of the the workspace and optimal path plan. The translucent circle directly corresponds to the sensor FoV from (a).

point and by Δp the distance between adjacent grid points. Without loss of generality we consider "4-way" adjacency of points (i.e., adjacent points are top, down, left, and right). The proposed methods and analyses remain unchanged if "8-way" adjacency (i.e., including points top-left, top-right, etc.) were considered.

We define a *threat field* $c : \mathcal{W} \to \mathbb{R}_{>0}$ as a strictly positive temporally static scalar field. We are interested in a path-planning problem of minimizing the agent's threat exposure. A *path* $\boldsymbol{\pi} = (\boldsymbol{\pi}[0], \boldsymbol{\pi}[1], \ldots, \boldsymbol{\pi}[\lambda])$ between prespecified initial and goal vertices $v_0, v_{\mathrm{L}} \in V$ is a finite sequence, without repetition, of successively adjacent vertices such that $\boldsymbol{\pi}[0] = v_0$ and $\boldsymbol{\pi}[\lambda] = v_{\mathrm{L}}$ for some $\lambda \in \mathbb{N}$. In a minor abuse of notation and when the meaning is clear from the context, we will denote by $\boldsymbol{\pi}$ the (unordered) set of vertices in a path. A *path incidence vector* $\boldsymbol{v}_{\boldsymbol{\pi}} \in \{0,1\}^{N_{\mathrm{g}}}$ has $\boldsymbol{v}_{\boldsymbol{\pi}}[i] = 1$ if $i = \boldsymbol{\pi}[j]$ for $j \in \{\lambda\}\backslash 0$ and $\boldsymbol{v}_{\boldsymbol{\pi}}[i] = 0$ otherwise. The *cost* of a path $\boldsymbol{\pi}$ is the total threat exposure calculated as $\mathcal{J}(\boldsymbol{\pi}) := \Delta p \sum_{j=1}^{\lambda} c(\boldsymbol{p}_{\boldsymbol{\pi}_j})$. The main problem of interest is to find a path $\boldsymbol{\pi}^*$ of minimum cost.

We cannot solve this problem as stated because the threat field is unknown. The threat field can be observed by a network $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{N_{\mathrm{s}}}\}$ of $N_{\mathrm{s}} \in \mathbb{N}$ sensors. Each of these sensors measures the threat in a circular FoV as shown in Figure 2.2. The center and radius of this circular FoV $\mathcal{F}_k \subset \mathbb{R}^2$, denoted $\boldsymbol{s}_k \in \mathcal{W}$ and $\varrho_k \in \mathbb{R}_{>0}$ for the k^{th} sensor, are parameters that we may

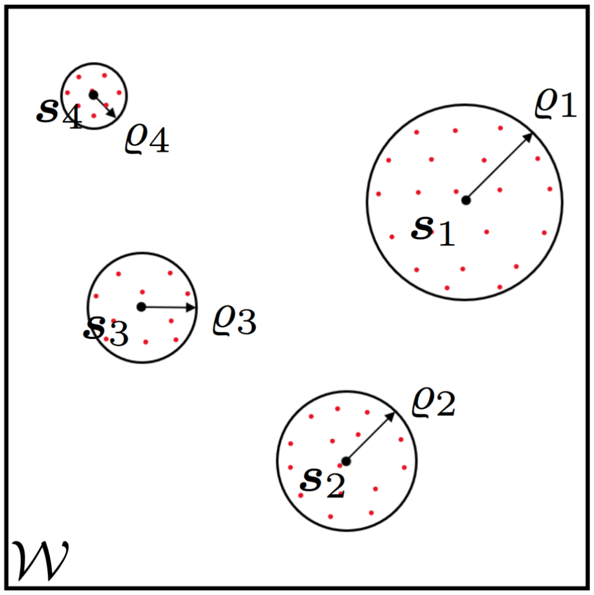


Figure 2.2: Illustration of sensor configuration for 4 sensors.

choose for each $k \in \{N_\mathrm{s}\}$. Maximum and minimum FoV radius constraints are specified as $\varrho^{\max}$ and $\varrho^{\min}$, respectively. The set of all sensor parameters is called a *configuration*, which we denote by $\mathcal{C} = \{\boldsymbol{s}_1, \varrho_1, \boldsymbol{s}_2, \ldots, \varrho_{N_\mathrm{s}}\}$.

Consider the set $\mathcal{F}_k \subset V$ of vertices with grid points within the FoVs of the k^{th} sensor. A sensor *cover incidence vector* $\boldsymbol{\nu}_k$ is defined such that $\boldsymbol{\nu}_k[i] = 1$ if $i \in \mathcal{F}_k$ and $\boldsymbol{\nu}_k[i] = 0$ otherwise. The cover incidence of all sensors in the network is $\boldsymbol{\nu} := (\boldsymbol{\nu}_1 \vee \boldsymbol{\nu}_2 \vee \ldots \vee \boldsymbol{\nu}_{N_\mathrm{s}})$. The k^{th} sensor takes $M_k \in \mathbb{N}$ pointwise and noisy measurements within its FoV $\mathcal{F}_k$. Each of these measurements is modeled as $z_{km} = c(\boldsymbol{x}_{km}) + \eta_{km}$, where $k \in \{N_\mathrm{s}\}$ and $m \in \{M_k\}$. Assumption 1 details the monotonicity assumption of the sensor noise.

Assumption 1. *The measurement error η_{km} is independent and identically distributed (i.i.d) with $\eta_{km} \sim \mathcal{N}(0, \sigma_k^2)$, where σ_k^2 is a monotonically increasing function of each sensor's FoV radius ϱ_k.*

We denote by $\boldsymbol{z} = [z_{11} \ldots z_{1M_1} \ldots z_{N_\mathrm{s}M_{N_\mathrm{s}}}]^\intercal$ the measurements made by the collection of all sensors. We may use these measurements to construct a stochastic estimate of the threat field, and in turn, to find an optimal path that minimizes the *expected* cost.

Furthermore, we would like the uncertainty in the estimated path cost to be low, e.g., we may require that the variance of the path cost be less than a certain prespecified threshold. To do so, we must collect a sufficient number of measurements, which may be achieved by repeatedly changing the sensor configuration over multiple iterations. Conceptually, at each iteration $\ell = 0, 1, \ldots, L$, the sensor configuration $\mathcal{C}_\ell^*$ is chosen, the threat field estimate is updated using the new measurements, and an optimal path may be computed. The main problem of interest is now reformulated as follows.

Problem 1. *Over a finite number of iterations $\ell = 0, 1, \ldots, L$, find sensor configurations $\mathcal{C}_\ell$ and a path $\boldsymbol{\pi}^*$ of minimum expected cost $\overline{\mathcal{J}}^* := \mathbb{E}[\mathcal{J}(\boldsymbol{\pi}^*)]$ that satisfies $\mathbb{E}[(\mathcal{J}(\boldsymbol{\pi}^*) - \overline{\mathcal{J}}^*)^2] \leqslant \varepsilon$.*

2.2 Coupled Sensor Configuration and Path-Planning

The proposed method, Coupled Sensor Configuration and Path-Planning (CSCP), is an iterative algorithm which directly couples optimization of the sensing and path-planning tasks. The pseudocode for CSCP is outlined in Figure 2.3. It involves three main components each iteration: (1) sensor configuration and data collection, (2) generation of a statistical threat field estimate from the data obtained from sensor configuration, and (3) determining the estimated optimal path-plan given the result of the field estimation. These stages iterate until the estimated path cost variance falls below a user specified path optimality threshold. We calculate the estimated path cost in Equation 2.1 as the vector product between the path incidence vector $\boldsymbol{v}_{\boldsymbol{\pi}}$ and an estimated threat field vector $\boldsymbol{f}$, which we describe in Section 2.2.3, scaled by the path transition cost Δp.

$$\overline{\mathcal{J}}_\ell(\boldsymbol{\pi}) = \mathbb{E}[\mathcal{J}(\boldsymbol{\pi})] = \Delta p \, \boldsymbol{v}_{\boldsymbol{\pi}}^\intercal \boldsymbol{f}_\ell \tag{2.1}$$

Similarly, we define the estimated path cost variance in Equation 2.2, where $\boldsymbol{P}_\ell$ is the current iteration estimated threat error covariance matrix.

$$\mathrm{Var}_\ell(\boldsymbol{\pi}) := \mathbb{E}[(\mathcal{J}(\boldsymbol{\pi}) - \overline{\mathcal{J}}_\ell(\boldsymbol{\pi}))^2] = (\Delta p)^2 \boldsymbol{v}_{\boldsymbol{\pi}}^\intercal \boldsymbol{P}_\ell \boldsymbol{v}_{\boldsymbol{\pi}} \tag{2.2}$$

The algorithm proceeds to iterate over the the aforementioned stages until the estimated path cost variance converges below a termination threshold. Herein, we provide the details of each stage of the CSCP algorithm.

Coupled Sensor Configuration and Path-Planning

```
1: Let $\ell := 0$, $\boldsymbol{f}_0 := \mathbf{0}$, $\boldsymbol{P}_0 := \chi \boldsymbol{I}$
2: Solve for $\boldsymbol{\pi}_0^* := \arg\min \overline{\mathcal{J}}_0(\boldsymbol{\pi})$
3: while $\mathrm{Var}_\ell(\boldsymbol{\pi}_\ell^*) > \varepsilon$ do
4:     Optimize Sensor Configuration $\mathcal{C}_\ell^*$
5:     Record new measurements $\boldsymbol{z}$
6:     Increment iteration counter $\ell := \ell + 1$
7:     Find GPR-based threat field estimate $\boldsymbol{f}_\ell$ and error covariance $\boldsymbol{P}_\ell$
8:     Find $\boldsymbol{\pi}_\ell^* := \arg\min \overline{\mathcal{J}}_\ell(\boldsymbol{\pi})$
```

Figure 2.3: Pseudocode for the CSCP algorithm to solve Problem 1.

2.2.1 Algorithm Initialization

The algorithm initializes "optimistically" by setting $\boldsymbol{f}_0 = \mathbf{0}$. The initial uncertainty in the threat estimate is quantified by initializing the estimation error covariance matrix with uniformly large values, e.g., $\boldsymbol{P} = \chi \boldsymbol{I}_{(N_{\mathrm{g}})}$, where $\chi \gg 1$ is an arbitrary large number. Due to this "optimistic" initialization, the initial optimal path $\boldsymbol{\pi}_0^*$ is of minimum length as $\boldsymbol{\pi}_0^* := \arg\min \overline{\mathcal{J}}_0(\boldsymbol{\pi})$, optimized using Dijkstra's algorithm. Upon initializing the initial threat estimate vector and threat error covariance matrix, we proceed to the iterative phase of the algorithm. Line 3 initializes the iterative phase whenever the estimated path cost variance is greater than a prespecified termination threshold ε. We begin each iteration by optimizing the set of sensor configuration parameters $\mathcal{C}_\ell$ which describe their field of view as detailed in the next section.

2.2.2 Sensor Network Configuration

Each iteration, after determining the path plan region of interest, we perform sensor configuration. This stage is responsible for configuring sensor locations and FoVs which best observe a set *region of interest* $\mathcal{R}_\ell \subset V$ at any iteration ℓ. The region of interest set $\mathcal{R}$ can be defined by a region of interest vector $\boldsymbol{r} \in \{0, 1\}^{N_{\mathrm{g}}}$. We can set the region of interest equal to the path incidence vector $\boldsymbol{r} = \boldsymbol{v}_{\boldsymbol{\pi}}$ each iteration, but in later chapters we will explore situations in which we set $\boldsymbol{r}$ differently. To quantify the sensor FoV coverage with respect to $\boldsymbol{r}$, we utilize a task-driven information gain

(TDIG) metric which represents the information gain via variance reduction of $\boldsymbol{r}$:

$$\tau(\mathcal{C}_\ell, \boldsymbol{r}) := \boldsymbol{r}^\mathsf{T}(\boldsymbol{P}_\ell - \boldsymbol{P}_{\ell+1})\boldsymbol{r} \tag{2.3}$$

The sensor configuration problem is of finding a configuration $\mathcal{C}_\ell^*$ that maximizes the TDIG subject to the constraints $\boldsymbol{s}_k \in \mathcal{W}$, $\varrho^{\min} \leqslant \varrho_k \leqslant \varrho^{\max}$, for each $k \in \{N_\mathrm{s}\}$. The formulation of the TDIG metric in Equation 2.3 cannot be directly solved as written due to the unavailability of $\boldsymbol{P}_{\ell+1}$. As such, an approximate posterior threat error covariance matrix $\hat{\boldsymbol{P}}_{\ell+1}$ must be formulated. Next, we present two alternatives for this approximation.

Fixed Correlations Approximation

The first approach to computing the approximate posterior threat covariance matrix $\hat{\boldsymbol{P}}_{\ell+1}$ is to fix the correlations between vertices. We begin by computing $\boldsymbol{Q} = diag(\boldsymbol{q})$, where $\boldsymbol{q}$ is a reduction factor vector for each point in the workspace and is computed with Equation 2.4.

$$\boldsymbol{q}^{-1} := diag(\boldsymbol{P}_\ell)^{-1} + \sum_{k=1}^{N_s} \boldsymbol{\nu}_k / \sigma_k^2 \tag{2.4}$$

In Equation 2.4, $diag(\boldsymbol{P}_\ell)$ represents the diagonal elements of the threat error covariance matrix for the current iteration. The reduction factor vector $\boldsymbol{q}$ is representative of the weighted combination of sensor noise and vertex uncertainty. After obtaining $\boldsymbol{Q}$, we can estimate the approximate posterior threat error covariance matrix by Equation 2.5.

$$\hat{\boldsymbol{P}}_{\ell+1} := \boldsymbol{Q}\boldsymbol{\Omega}_\ell\boldsymbol{Q} \tag{2.5}$$

The variable $\boldsymbol{\Omega}_\ell$ represents the current iteration threat error correlation matrix, which is directly obtainable from the current iteration threat error covariance matrix $\boldsymbol{P}_\ell$. This approximation method preserves the latent correlations between workspace vertices, but when optimizing the TDIG with this approximation, it is prone to overestimate the TDIG metric due to the positive correlations between vertices which may not be preserved when new data is collected.

Independent Vertices Approximation

Alternatively, we may ignore the correlation and assume independence of the workspace vertices. In contrast, this is an admissible approach as we do not overestimate the posterior threat error covariance matrix. We can therefore take $\hat{\boldsymbol{P}}_{\ell+1} = \boldsymbol{Q}$ and find a sensor network configuration $\mathcal{C}_\ell^*$ that maximizes the TDIG along $\boldsymbol{r}$, subject to sensor constraints, that does not overestimate sensor cover performance between spatially distant workspace vertices.

2.2.3 GPR-based Field Estimation

Upon optimizing for sensor configuration $\mathcal{C}_\ell^*$ as previously described and obtaining measurements $\boldsymbol{z}$ within each sensor's FoV, we can utilize this data for estimating the threat field. To perform the threat field estimation, we utilize Gaussian Process Regression (GPR). GPR is a nonparametric supervised learning method which is a stochastic process of random variables. Gaussian Processes make use of a kernel which defines the relationship between training data points. For a set of training points $\boldsymbol{X} = \{\boldsymbol{x}_{11}, \ldots, \boldsymbol{x}_{km}, \ldots, \boldsymbol{x}_{N_\mathrm{s}M_{N_\mathrm{s}}}\}$, we can define a matrix kernel $\boldsymbol{K} \in \mathbb{R}^{M \times M}$ using anisotropic radial basis functions (RBF-ARD) as:

$$\boldsymbol{K}[i,j] = \kappa(\boldsymbol{X}, \boldsymbol{X}) := \exp(-\tfrac{1}{2}(\boldsymbol{x}_i - \boldsymbol{x}_j)^\mathrm{T} \boldsymbol{\Theta}_r^{-2} (\boldsymbol{x}_i - \boldsymbol{x}_j)) \tag{2.6}$$

Likewise, $\boldsymbol{\Theta}_r$ is a matrix of input dimension length scales. Along the diagonal are the dimension specific length scales and the off-diagonal are the length scale correlations between dimensions. One important property of this kernel is it has a covariance matrix structure equivalent to it's correlation matrix. We will denote the RBF-ARD kernel as $\boldsymbol{K}^R$. It is often common to add a scaling kernel $\boldsymbol{K}^S := \theta_c \boldsymbol{K}$ where θ_c is a scaling parameter to be chosen. In this work we use a kernel formulated as $\boldsymbol{K} = \boldsymbol{K}^S \boldsymbol{K}^R$.

For a set of training points $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{km}, \ldots, \boldsymbol{x}_{N_\mathrm{s}M_{N_\mathrm{s}}}\}$ and a set of test points $\boldsymbol{X}_*$, we can define kernels $\boldsymbol{K}_* = \kappa(\boldsymbol{X}, \boldsymbol{X}_*)$ and $\boldsymbol{K}_{**} = \kappa(\boldsymbol{X}_*, \boldsymbol{X}_*)$ in a similar manner. The choice of kernel of RBF-ARD is a good generic kernel choice, but it can be replaced with alternative kernels depending on the application and function that is being modeled. In later chapters, we make modifications to the GPR formulation to accommodate special scenarios.

The hyperparameters $\boldsymbol{\theta} = (\theta_c, \boldsymbol{\Theta}_r)$ are found by minimizing the negative marginal log-likelihood:

$$\log p(\boldsymbol{z}|\boldsymbol{X}, \boldsymbol{\theta}) = -\frac{1}{2}\left(\boldsymbol{z}^\intercal \boldsymbol{K}_z^{-1} \boldsymbol{z} + \log|\boldsymbol{K}_z| + N_{\text{meas}} \log 2\pi\right) \tag{2.7}$$

We define the kernel with additional heteroscedastic noise as $\boldsymbol{K}_z := \boldsymbol{K} + diag(\boldsymbol{\sigma})$, where $\boldsymbol{\sigma} := \begin{bmatrix}\sigma_1^2 & \dots & \sigma_{km}^2 & \dots \sigma_{N_s M_{N_s}}^2\end{bmatrix}^\intercal$ for $m \in M_k$ and $k \in N_s$. The joint distribution of observations $\boldsymbol{z}$ and the mean threat $\boldsymbol{f}$ is given by:

$$\begin{bmatrix}\boldsymbol{z} \\ \boldsymbol{f}\end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix}\boldsymbol{K_z} & \boldsymbol{K}_* \\ \boldsymbol{K}_*^\intercal & \boldsymbol{K}_{**}\end{bmatrix}\right) \tag{2.8}$$

From the joint distribution, we can obtain the current iteration threat field estimate and threat error covariance matrix as:

$$\boldsymbol{f}_\ell = \boldsymbol{K}_*^\intercal \boldsymbol{K}_{\boldsymbol{z}}^{-1} \boldsymbol{z} \tag{2.9}$$

$$\boldsymbol{P}_\ell = \boldsymbol{K}_{**} - \boldsymbol{K}_*^\intercal \boldsymbol{K}_{\boldsymbol{z}}^{-1} \boldsymbol{K}_* \tag{2.10}$$

2.2.4 Termination Criteria

After successively finding the optimal sensor configuration, collecting sensor measurements, and computing a new statistical threat field estimate, we are able to evaluate for a new estimated optimal path-plan $\boldsymbol{\pi}_\ell^* := \arg\min \overline{\mathcal{J}}_\ell(\boldsymbol{\pi})$. As for the initial path-plan estimate, we may use Dijkstra's algorithm to find the optimal sequence of vertices. The algorithm breaks from it's loop if the new path-plan $\boldsymbol{\pi}_\ell^*$ is such that $\mathrm{Var}_\ell(\boldsymbol{\pi}_\ell^*) \leqslant \varepsilon$. We note that the smaller the value of ε, the higher the desired confidence in our path-plan and vice versa.

2.2.5 Algorithm Properties and Convergence

Proposition 1. *The CSCP algorithm terminates in a finite number of iterations $L \in \mathbb{N}$ for $N_{\text{s}} > 0$.*

Proof. At any iteration $\ell \in \mathbb{N}$, by Line 4 of Algorithm 2.3, each sensor is configured such that its FoV intersects with the current iteration ROI $\mathcal{R}_\ell$. By any definition of the ROI, $\boldsymbol{\pi}_\ell^* \subset \mathcal{R}$ and therefore, at least one measurement is taken on at least one vertex of path $\boldsymbol{\pi}_\ell^*$ whenever $N_\mathrm{s} > 0$. The path cost variance monotonically decreases at each iteration as $\mathrm{Var}_{\ell+1}(\boldsymbol{\pi}_\ell^*) < \mathrm{Var}_\ell(\boldsymbol{\pi}_\ell^*)$,. Because the path cost variance is also bounded below by zero, $\inf_\ell\{\mathrm{Var}_\ell(\boldsymbol{\pi}_\ell^*)\} = 0$, it follows that there exists a finite $L \in \mathbb{N}$ such that $\mathrm{Var}_L(\boldsymbol{\pi}_L^*) < \varepsilon$, which is when the algorithm terminates. □

Proposition 2. *The CSCP algorithm solves Problem 1.*

Proof. We note that the path $\boldsymbol{\pi}_L^*$ satisfies the conditions stated in Problem 1, namely, that is has minimum expected cost $\overline{\mathcal{J}}^* = \overline{\mathcal{J}}_L(\boldsymbol{\pi}_L^*)$ and $\mathbb{E}[(\mathcal{J}_L - \overline{\mathcal{J}}^*)^2] = \mathrm{Var}_L(\boldsymbol{\pi}_L^*) < \varepsilon$ per the termination criterion enforced by Line 3 in Algorithm 2.3. □

Corollary 1. *The path $\boldsymbol{\pi}_L^*$ is near-optimal in the following sense:*

$$\mathbb{P}\left[|\overline{\mathcal{J}}^* - \mathcal{J}_L| \leqslant 3\sqrt{\varepsilon}\right] \geqslant 0.9973.$$

Proof. Due to the GPR-based field estimation and the linearity of $\mathcal{J}(\boldsymbol{\pi})$, the path cost is normally distributed, and the result follows immediately from the standard normal table. □

Corollary 2. *The path $\boldsymbol{\pi}_L^*$ is near-optimal in the following sense. Let $\mathcal{J}^*$ denote the cost of the true optimal path. Then:*

$$\mathbb{P}\left[|\mathcal{J}_L - \mathcal{J}^*| \leqslant 3\sqrt{\varepsilon}\right] \geqslant 0.9973.$$

Proof. Due to the GPR-based field estimation and the linearity of $\mathcal{J}(\boldsymbol{\pi})$, the path cost is normally distributed, and the result follows immediately from the standard normal table. □

2.2.6 Results and Discussion

In this section, we provide sample results of numerical simulations of the CSCP algorithm and a comparison with traditional information-maximizing (Info-Max) approaches. The Info-Max approach finds sensor configurations to maximize the reduction of $\mathrm{tr}(\boldsymbol{P}_{\ell-1})$. Compared to the proposed sensor placement based on the TDIG, the Info-Max approach is entirely decoupled from the path-planning problem and attempts to estimate the field with uniformly low error in all spatial regions. We note that such an approach is the current state of the art if one were to attempt a sensor cover and placement problem as we have defined. While many methods exist for pointwise sensor placement and area coverage, there does not exist a method for the combination of the two as we have defined for task-driven information gain.

We conducted a study with randomly generated threat fields $c(\boldsymbol{x}) = \sum_{n=1}^{N_p} \theta_n \phi_n(\boldsymbol{x})$, where θ_n are randomly chosen coefficients, and ϕ_n are radial basis functions that cover $\mathcal{E}$. The number of bases N_p is indicative of the "richness" in spatial variations in the threat field: fields with small N_p have a few peaks and several flat regions whereas fields with very large N_p have closely spaced peaks, which may cause several flat regions.

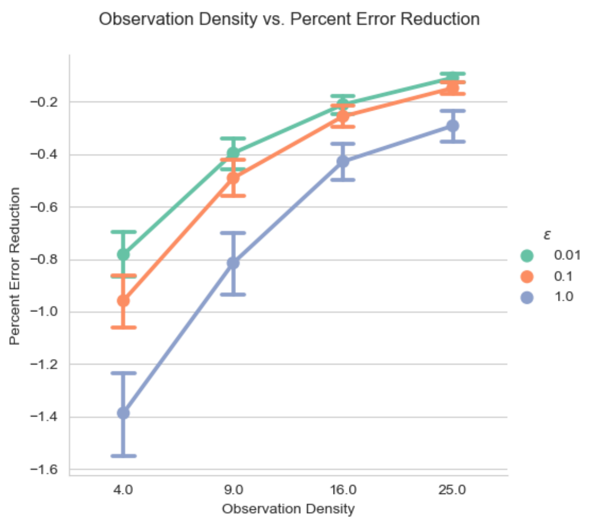
Parameter	Set of simulated values
# basis functions N_p	$\{8, 16, 32, 64, 128, 256, 512, 1024\}$
# sensors N_s	$\{1, 2, 4, 9, 16\}$
# grid points N_g	$\{11^2, 21^2, 31^2, 41^2, 51^2\}$
# points in sensor FoV M_k	$\{4, 9, 16, 25\}$, for each $k \in [N_\mathrm{s}]$
Termination threshold ε	$\{0.01, 0.1, 1.0\}$

Table 2.1: Set of parameters used in CSCP vs. Info-Max numerical performance analysis.

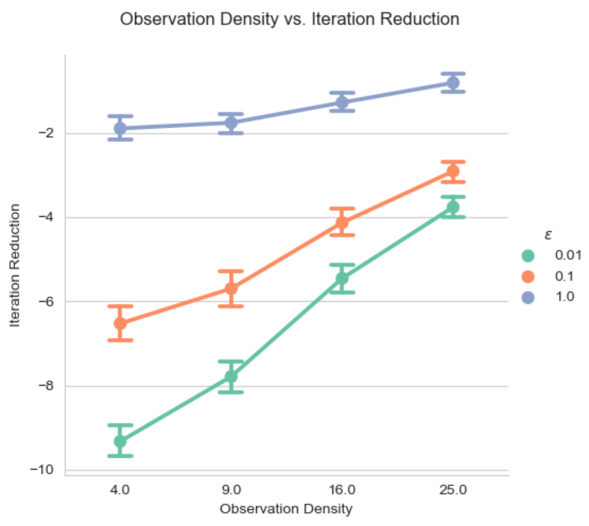
100 random trials were conducted for each combination of the parameters listed in Table 2.1, for a total of 240,000 simulations. In what follows, we describe the impact of each of these parameters on the average path cost suboptimality and on the number of iterations required for convergence. We compare the performance of the CSCP algorithm in these simulations to Info-Max sensor placement, which attempts to reduce the uncertainty of the entire workspace. Additional plots for all of averaged experiment parameters for CSCP and Info-Max are made available in the Appendix D, however we summarize their findings herein. Note that negative numbers indicate *improvement* by using CSCP.

Observation Density

We begin by assessing the suboptimality of the paths and the average iterations required for convergence for each option of observation density. We directly take the difference between CSCP and Info-Max to obtain the reduction in percent error and iterations. Figure 2.4 depicts both plots showing respective reduction by using CSCP in place of Info-Max. We note that the CSCP method not only has lower path optimality error but also increasingly fewer required iterations as the observations taken by each sensor becomes sparser. Such a trend is significant and embodies the notion that CSCP can find near-optimal path-plans with *far fewer* observations than Info-Max strategies.



(a)



(b)

Figure 2.4: Average percent error (a) and iteration (b) reduction by using CSCP in place of Info-Max for various observation densities.

Environment Complexity

Next, we assess the performance in terms of complexity, or richness, of the threat environment. Richness is in contrast to low threat parameter counts which are sparse (many valleys) or high threat parameter counts which saturate the threat environment (many plateaus). We see by assessing Figure 2.5 (a) that the error reduction by using CSCP in place of Info-Max is maximum at the 'richest' number of parameters. Figure 2.5 (b) shows that the CSCP algorithm has near constant performance improvements regardless of threat value which is not too sparse. Yet, even sparse threat environments yield iteration reduction by using CSCP.

Number of Available Sensors

The number of sensors available for configuration yielded more positive results for CSCP in place of Info-Max. Figure 2.6 (a) shows that the CSCP method achieves increasingly better

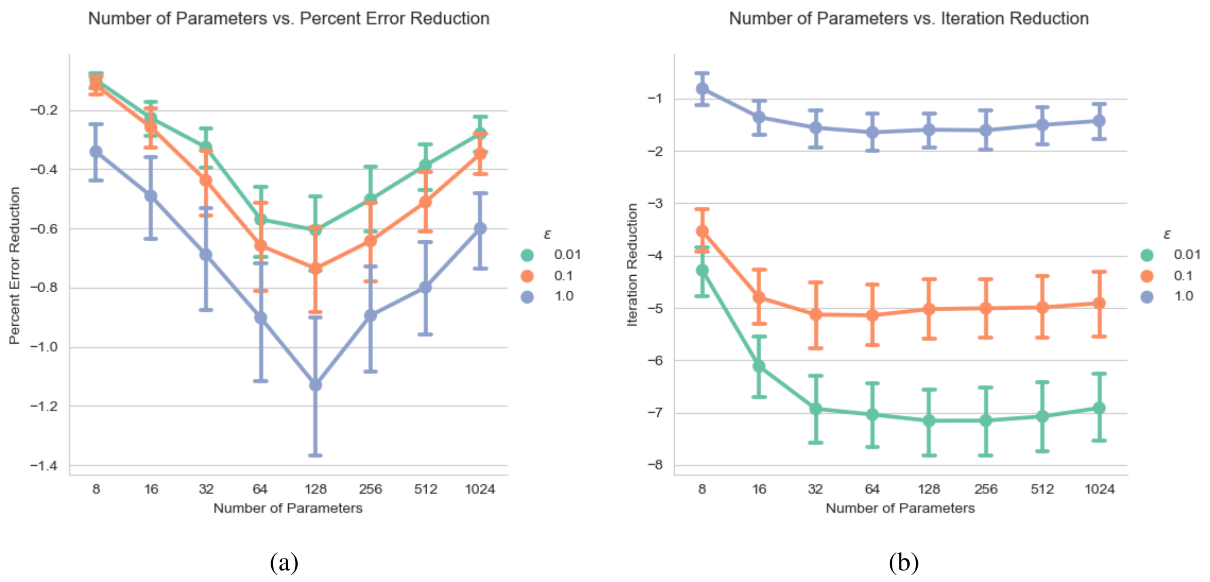


Figure 2.5: Average percent error (a) and iteration (b) reduction by using CSCP in place of Info-Max for various numbers of threat parameters.

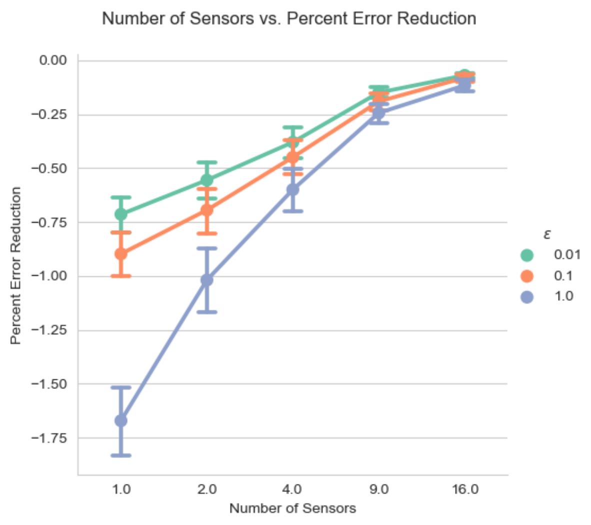
performance as the number of sensors decreases. While Figure 2.6 is a nonlinear relationship, the general trend follows that the iteration reduction increases in performance as fewer sensors are available. This leads to the conclusion that CSCP can find near-optimal path-plans in fewer iterations with low sensor counts.

Workspace Resolution

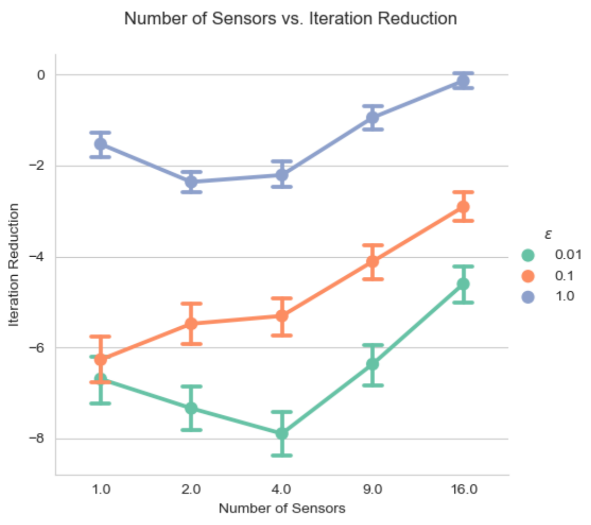
The workspace resolution was another parameter used for comparison between CSCP and Info-Max. As shown in Figure 2.7, the CSCP algorithm was more proficient in terms of iterations than Info-Max with increasing resolution. It did not seem to play a significant part in the error reduction. The takeaway is that the CSCP algorithm can make better use of information stored at finer grid resolutions, accelerating convergence.

Termination Threshold

Lastly, we varied the termination threshold with values $\varepsilon \in \{1, 0.1, 0.01\}$ which are in order of increasing path-optimality requirements. We note that the results yielded that the CSCP algorithm kept average suboptimality below 1.0% for any value of N_p with $\varepsilon = 1$ and below 0.5% with



(a)



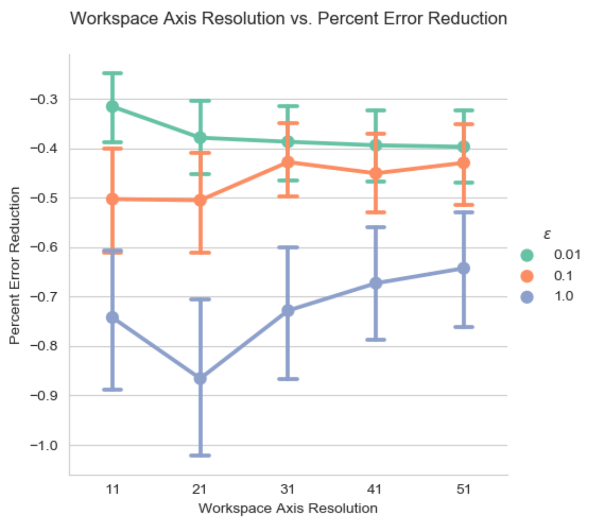
(b)

Figure 2.6: Average percent error (a) and iteration (b) reduction by using CSCP in place of Info-Max for various numbers of available sensors.

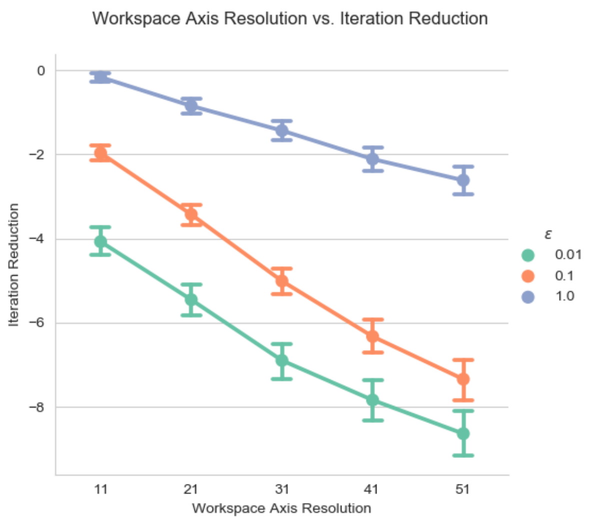
$\varepsilon = 0.01$. Across all experiments, as well as those in Appendix D, we show that the CSCP algorithm performance increasingly outperforms the Info-Max approach as the termination threshold becomes more strict. We attribute this to the exploitative nature of sensor configuration to vertices within the region of interest which drive down path uncertainty faster than vertices outside the region of interest.

Results Summary

We found the average path-plan suboptimality to be 0.436%, which supports the claim that CSCP provides near-optimal paths. The average number of iterations L over all simulations for the algorithm to terminate is 6.88, which supports the claim that the CSCP algorithm is computationally efficient. Expanded heatmaps of the performance of CSCP in place of Info-Max for the parameters are shown in Figures D.1, D.2, and D.3. The average total number of iterations, over all simulations, required by the proposed algorithm is 38.36% less than that required by the Info-Max approach. In summary, the CSCP algorithm finds near-optimal paths with significantly fewer measurements. Note that the total number of sensor measurements is $L\sum_{k=1}^{N_s} M_k$; i.e., the total number of measurements is proportional to the number of iterations. In certain cases, the number of measurements required is reduced by *an order of magnitude*.



(a)



(b)

Figure 2.7: Average percent error (a) and iteration (b) reduction by using CSCP in place of Info-Max for various grid resolutions.

2.3 Disaster Scenario in St. Lucia

To help better visualize the operation of this algorithm, consider a fictional disaster scenario on the island of St. Lucia in Figure 2.8. In this hypothetical scenario, major roadways are flooded or otherwise incapacitated preventing normal travel over the island. Survivors, which are located at the Fox Grove Inn (bottom right marker), need to be routed to the Hewannora International Airport (top left marker) for extraction. The challenge however, is that unknown threats (flooded regions, downed power lines, etc.) exist and need to be avoided.

Consider that a network of mobile aerial vehicles, $N_s = 3$, equipped with exteroceptive visual surveying equipment are available to observe the extent of hazardous elements on the island. Using this data, a transport path for the survivors can be provided. In this scenario, the amount of time spent collecting data must be kept to a minimum. We now show how the CSCP algorithm may be applied to solve such a scenario. The problem is performed with a workspace resolution $N_g = 41^2$ (approximately 0.4 km spacing) with a termination threshold of of $\varepsilon = 1$.

The algorithm begins by finding the shortest path with Dijkstra's algorithm from the start to goal vertex. Figure 2.9 shows the true optimal path (white) and the estimated optimal path (green), which is this shortest distance path. The equally spaced dots represent the collection of grid space

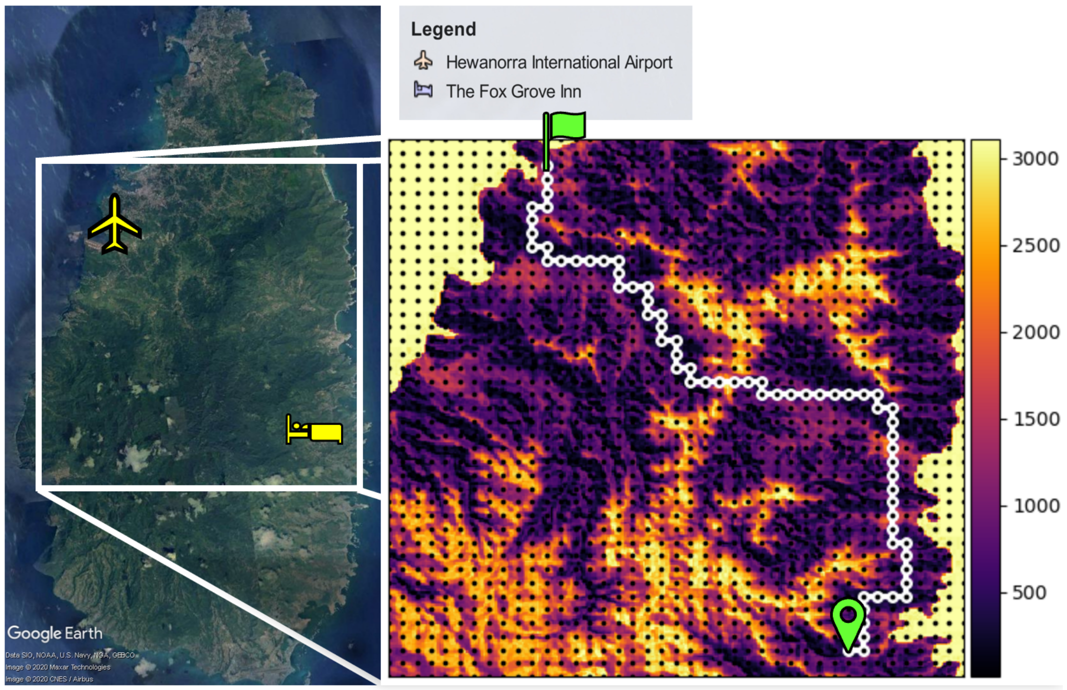


Figure 2.8: Example workspace and true optimal path the marker indicator to the goal location denoted by the flag in the St. Lucia disaster relief scenario.

vertices in which the agent (the St. Lucia survivors) can traverse over. Note that the initial path-plan estimate is taken as the shortest path due to the optimistic field estimate assumption $\boldsymbol{f}_0 = \boldsymbol{0}$ and the pessimistic threat error covariance $\boldsymbol{P}_0 = \chi \boldsymbol{I}$, which enforce maximum uncertainty over the workspace.

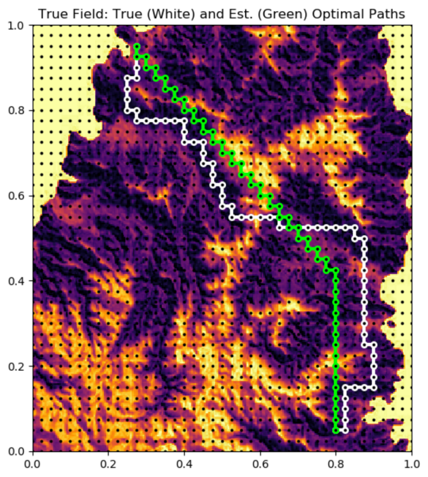


Figure 2.9: St. Lucia workspace (black dots), threat field, and true (white) and estimated (green) optimal path-plans at the initial iteration $\ell = 0$.

After determining the initial estimated path-plan $\boldsymbol{\pi}_0^*$ we proceed to configure sensors with direct relevance to the path-plan. In Figure 2.10, the translucent circles depict the sensor cover which are determined by optimizing the TDIG metric given the prior estimated path-plan. Upon receiving the data collected from the sensors, we proceed to fit the GP model to produce the estimated threat field. Figure 2.10 shows that the new estimated path-plan skirts the outer boundary as a result of the estimated field being optimistic away from the training data. The path cost variance is computed at the end of the iteration and assessed to see if it satisfies our criteria for path optimality. Clearly, this path-plan passes through high uncertainty workspace vertices and as such does not pass our termination threshold criteria of $\varepsilon = 1$. We continuously perform these sequence of events until this criteria is satisfied. In what follows, we show the second, seventh, and fourteenth and final iteration of the St. Lucia problem in Figures 2.11, 2.12, and 2.13, respectively.

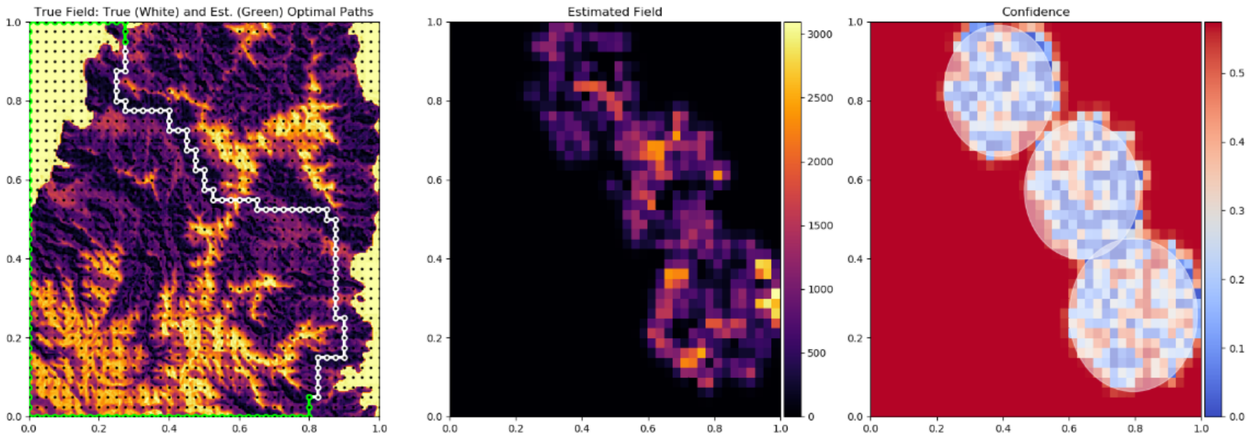


Figure 2.10: The first estimated path-plan, estimated threat field, and variance of each vertex with associated sensor coverage (translucent circles).

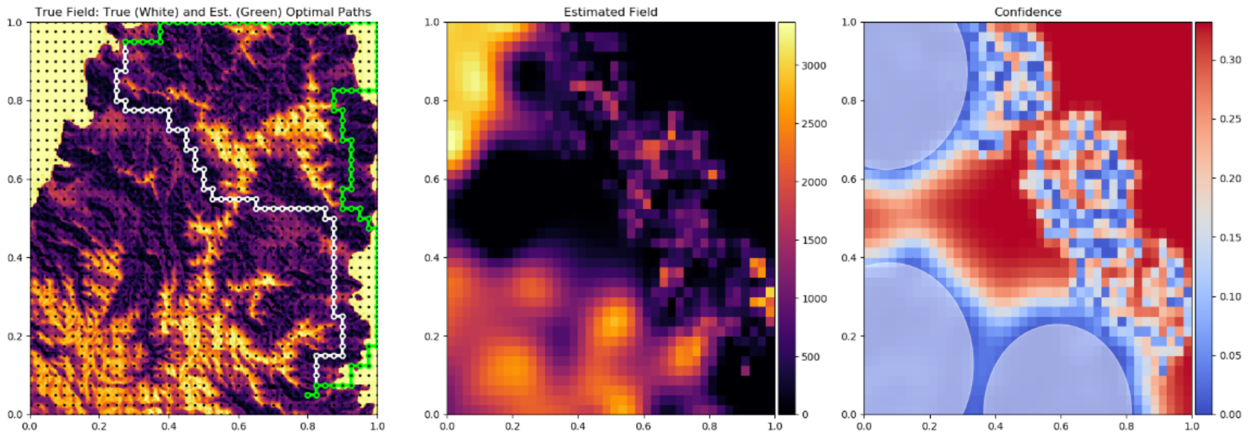


Figure 2.11: The second estimated path-plan, estimated threat field, and variance of each vertex with associated sensor coverage.

The final path-plan determined by CSCP in figure 2.13 is spatially close to the true optimal path-plan with only few deviations. Of note is the final threat field estimate granularity at various

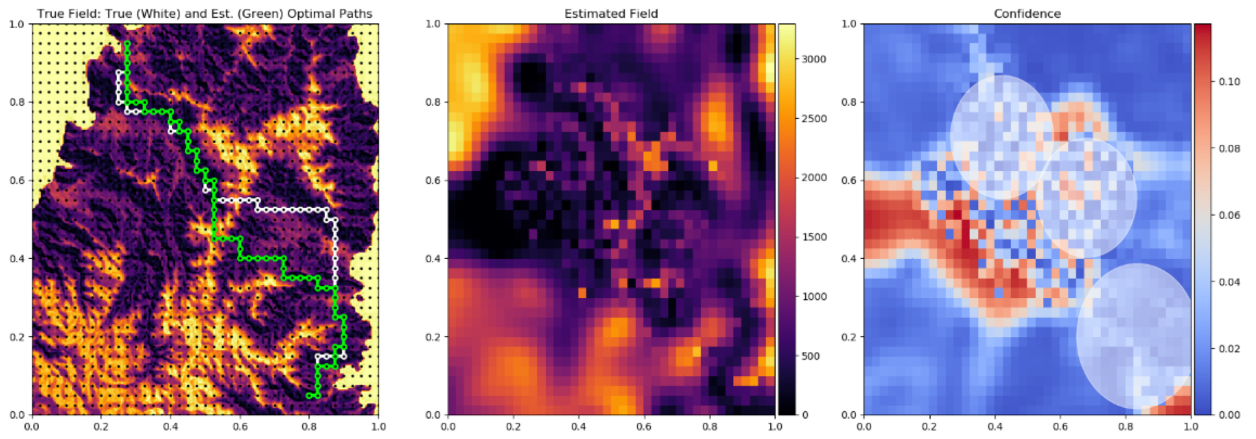
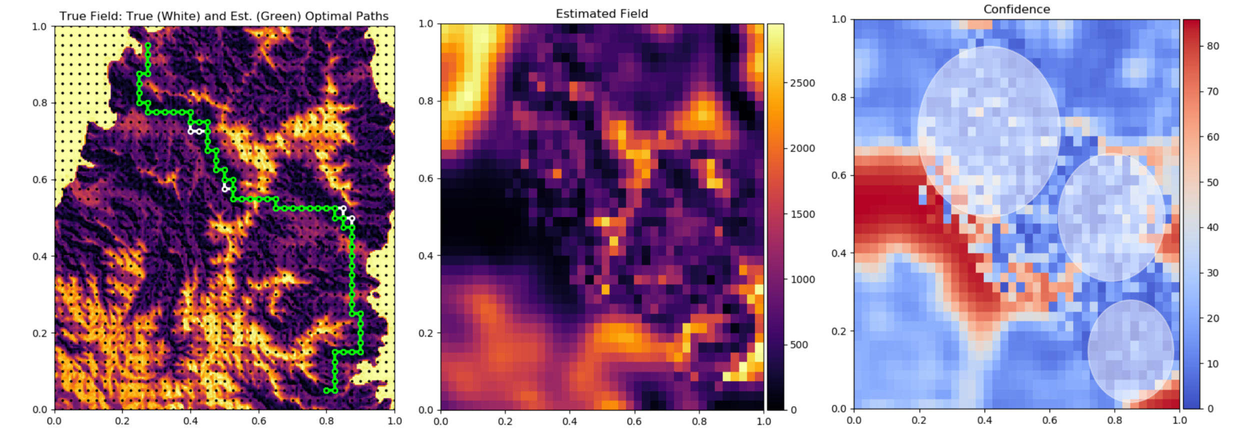


Figure 2.12: The seventh estimated path-plan, estimated threat field, and variance of each vertex with associated sensor coverage.



(a) (b) (c)

Figure 2.13: The final estimated path-plan, estimated threat field, and variance of each vertex with associated sensor coverage at $\ell = L$.

locations. Notice that the estimation error variance is high and the field does not capture threat variations in regions far away from the true path-plan. This is a characteristic of the TDIG metric, which enforces exploitation with sensor coverage of vertices with direct relevance to the planning efforts. While observations are made here due to the optimistic nature of the algorithm, further exploration is not performed as it is deemed an infeasible path-plan solution. In contrast, the areas in which more sensors are configured more accurately represents the true field.

The path cost begins with a suboptimality of 578% given our initial path plan and upon convergence achieves only 1.68% suboptimality for the St. Lucia example. Figure 2.14 shows the path cost and variance at each iteration until convergence at iteration $L = 14$. We note that the 'bump' in the path cost and variance indicates the exploratory nature of the algorithm due to the optimistic nature of the estimated field spatially distant from existing training data. In summary, the CSCP algorithm finds a near-optimal path-plan for the survivors to follow that is within the optimality criteria for safe transport to the airport.

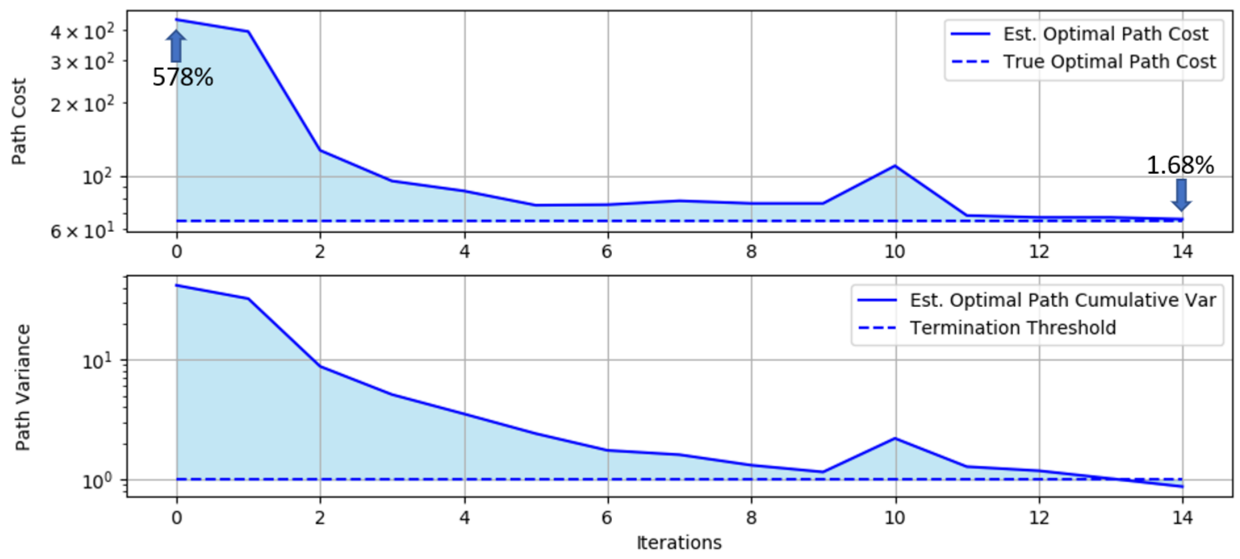


Figure 2.14: Path cost and variance values at each iteration for the St. Lucia scenario.

Chapter 3

Greedy Batched CSCP for Safety Critical Scenarios

3.1 Greedy Batched CSCP

The proposed algorithm provides an alternative methodology to the basic CSCP algorithm introduced in Chapter 2, which provided a solution that finds the optimal sensor configuration along the *entire* path each iteration. In situations where safety is of the upmost importance, the termination threshold ε may be set to an extremely small value. When the termination threshold ε is extremely strict, the CSCP algorithm can spend many iterations sampling the same area until the confidence in the path-plan converges below ε. This causes unnecessary computational expenditure to refit the statistical field estimate and determine a new estimated path-plan each iteration.

To remedy this issue for such safety critical scenarios, we present a Greedy Batched CSCP algorithm (GB-CSCP) which focuses on quickly arriving to a strict path optimality condition. Specifically, it differs from the algorithm of Chapter 2 by using a two stage approach: (1) *Exploration*, which focuses sensor FoV along unidentified vertices, and (2) *Exploitation*, which assumes that a fully identified path is the true optimal path. As such, it allows for batches of sensor configurations within a single iteration to drive the estimated path cost variance below a user specified termination threshold.

3.1.1 Greedy Region of Interest Pruning

The GB-CSCP algorithm operates by keeping track of a set of identified vertices $\mathcal{I} \in V$ which is initialized as $\mathcal{I} := \emptyset$. We classify a vertex identified whenever it is covered by a sensor FoV. As such, each iteration we update this set as $\mathcal{I} := \mathcal{I} \cup \{\cup_{k=1}^{N_{\mathrm{s}}} \mathcal{F}_k\}$. Likewise, an *identified incidence vector*, $\mathcal{I}_{\boldsymbol{\pi}}$, is defined as $\mathcal{I}_{\boldsymbol{\pi}} \in \mathbb{R}^{N_{\mathrm{g}}}$ such that $\mathcal{I}_{\boldsymbol{\pi}}[i] = 1$ if $i \in \mathcal{I}_{\boldsymbol{\pi}}[j]$ for $j \in [\lambda]\backslash 0$ and

Greedy Batched Coupled Sensor Configuration and Path-Planning

```
1:  Let $\ell := 0$, $\boldsymbol{f}_0 := \mathbf{0}, \boldsymbol{P}_0 := \chi \boldsymbol{I}, \mathcal{I} := \emptyset$
2:  Solve for $\boldsymbol{\pi}_0^* := \arg\min \overline{\mathcal{J}}_0(\boldsymbol{\pi})$
3:  while $\mathrm{Var}_\ell(\boldsymbol{\pi}_\ell^*) > \varepsilon$ do
4:    if $|\boldsymbol{\pi} \backslash \mathcal{I}| \geqslant N_\mathrm{s}$ then
5:      Set $\boldsymbol{r} := \boldsymbol{v}_{\boldsymbol{\pi}} - \mathcal{I}_{\boldsymbol{\pi}}$
6:      Optimize Sensor Configuration
7:    else
8:      Let $\hat{L} := \ell$, $\boldsymbol{r} := \boldsymbol{v}_{\boldsymbol{\pi}}$
9:      while $\mathrm{Var}_{\hat{L}}(\boldsymbol{\pi}_\ell^*) > \varepsilon$ do
10:       Increment batch counter $\hat{L} := \hat{L} + 1$
11:       Optimize Sensor Configuration with $N_\mathrm{s} := N_\mathrm{s}(\hat{L} - \ell)$
12:       Find $\mathrm{Var}_{\hat{L}}(\boldsymbol{\pi}_\ell^*)$ with $\hat{\boldsymbol{P}}_{\hat{L}}$
13:   Record new measurements $\boldsymbol{z}$
14:   Increment iteration counter $\ell := \ell + 1$.
15:   Find GPR-based threat field estimate $\boldsymbol{f}_\ell$ and error covariance $\boldsymbol{P}_\ell$
16:   Find $\boldsymbol{\pi}_\ell^* := \arg\min \overline{\mathcal{J}}_\ell(\boldsymbol{\pi})$
```

Figure 3.1: Pseudocode for the Greedy Batched CSCP algorithm to solve Problem 1.

$\mathcal{I}_{\boldsymbol{\pi}}[i] = 0$ otherwise. The set of identified vertices is used for determining the algorithm's sensor configuration optimization state (exploration versus exploitation).

The exploration stage occurs whenever there are more unidentified path vertices than there are sensors available, $|\boldsymbol{\pi} \backslash \mathcal{I}| \geqslant N_\mathrm{s}$. We perform optimize the sensor configuration with the TDIG metric τ with the setting of $\boldsymbol{r} := \boldsymbol{v}_{\boldsymbol{\pi}} - \mathcal{I}_{\boldsymbol{\pi}}$ to find an optimal sensor configuration $\mathcal{C}_\ell$.

The exploitation stage occurs whenever $\boldsymbol{\pi}_\ell^*$ has strictly less identified vertices than there are available sensors. During this stage, we exploit the ability to have batches of the number of available sensors take additional observations within a single iteration. We describe the batch process in the next section.

3.1.2 Batched Sensor Configuration

As mentioned, when we are in the exploitative state, we provide the ability to batch N_s. The motivation is to drive the estimated path cost variance below the termination threshold in a single iteration. To do so, we set an estimated terminal iteration variable to the current iteration $\hat{L} := \ell$

and set $\boldsymbol{r} := \boldsymbol{v}_{\boldsymbol{\pi}}$. Next, we recursively increment the estimated terminal iteration, optimize the sensor configuration with $N_{\mathrm{s}} := N_{\mathrm{s}}(\hat{L} - \ell)$, and compute the estimated path cost variance at the estimated terminal iteration. This recursion occurs until the estimated path cost variance is below the termination threshold ε. The desired effect is reducing the need to perform sensor observations, update the field estimate and error covariance and obtain a new estimated optimal path for the $\hat{L} - \ell$ batched sensor configurations. Depending on a sensor's signal-to-noise ratio and the specification of ε, a batched sensor configuration method could greatly reduce the number of iterations of the outer CSCP algorithm loop by exploiting multiple samples of the region of interest.

3.1.3 Algorithm Properties

In addition to the algorithm properties demonstrated in Chapter 2, we state the following additional items specific to the Greedy Batched CSCP algorithm.

Proposition 3. *The Exploitative loop of Line 9 of Algorithm 3.1 procedure is guaranteed to converge in a finite number of iterations with $N_s > 0$.*

Proof. We note that with $N_{\mathrm{s}} > 0$ at least one sensor covers a path vertex such that $\boldsymbol{\nu}$ has at least one nonzero entry. Additionally, each iteration that does not satisfy ε, we add another batch of N_{s} sensors. We note that the path cost variance is monotonically decreasing in this case and is bounded by 0 and the prior path cost variance $\mathrm{Var}_{\ell}(\boldsymbol{\pi}_{\ell}^{*})$, thus we can say by monotone convergence theorem we converge to $\inf\{\mathrm{Var}_{\hat{L}}(\boldsymbol{\pi}_{\ell}^{*})\} = 0 < \varepsilon$. □

3.2 Results and Discussion

To quantitatively assess the performance of the GB-CSCP algorithm, we performed a numerical study with randomly generated threat fields in the series form $c(\boldsymbol{x}) = \sum_{n=1}^{N_p} \theta_n \phi_n(\boldsymbol{x})$, where ϕ_n are radial basis functions that cover $\mathcal{E}$. Threat intensity θ_n values were set to 100 for all basis functions N_p. The number of bases N_p is indicative of the "richness" in spatial variations in the threat field: fields with small N_p have a few peaks and several flat regions whereas fields with very large N_p have closely spaced peaks, which may cause several flat regions. We set $N_g = 21^2$ for this experimentation. The results from Chapter 2 showed that the performance scales proportionally to the termination threshold, so for these experiments we consider a fixed termination threshold $\epsilon = 0.05$. To simulate the minimum and maximum heights for UAVs, we constrained the sensor FoV to $\varrho_{min} = 0.01$ km and $\varrho_{max} = 1$ km. The parameters used in this experiment are shown in Table 3.1 and the results are shown in Table 3.2. The key results of GB-CSCP are as follows.

Parameter	Set of simulated values
# basis functions N_p	$\{25, 50, 75, 100\}$
# sensors N_{s}	$\{1, 3, 5, 7, 9\}$
Environment Area $\lvert\mathcal{E}\rvert$	$\{1\text{km}^2, 4\text{km}^2, 9\text{km}^2, 16\text{km}^2, 25\text{km}^2\}$

Table 3.1: Set of parameters used in numerical performance analysis.

	Iterations	Observations	Est. Error %	Incurred Error %	Field ID %	Runtime [sec]
CSCP	10.58 ± 6.28	52.90 ± 31.40	0.31 ± 0.91	0.06 ± 0.25	76.24 ± 12.67	671.15 ± 774.23
GB-CSCP	10.17 ± 5.21	49.24 ± 32.24	0.33 ± 0.99	0.06 ± 0.26	76.27 ± 12.66	446.27 ± 506.17
Info-Max	49.16 ± 46.71	245.80 ± 28.71	0.58 ± 5.23	2.03 ± 41.33	96.73 ± 6.44	647.80 ± 669.26

Table 3.2: Average Results of the GB-CSCP Numerical Study

CSCP requires less environment knowledge than Info-Max for convergence

As the set of identified workspace vertices $\mathcal{I}$ were tracked at each iteration we took an average of the identified percentage of the environment for CSCP, GB-CSCP, and Info-Max. The field identification percentage, as shown in Table 3.2, show that the CSCP and GB-CSCP approximately identify the same percentage of the environment, but both nearly 20% less than Info-Max. This result emphasizes the coupled sensor configuration approach and embodies the notion that CSCP requires less knowledge about the environment to determine a near-optimal path-plan.

Greedy exploration and batching reduces iterations with strict termination criteria

The results we obtained, namely the average iterations from Table 3.2 show that the the GB-CSCP algorithm performs slightly fewer iterations on average than CSCP. We note that we reduce the number of observations by 3.66 on average. Notably however, is that as the termination criteria becomes increasingly strict, the gap in performance between the GB-CSCP algorithm and CSCP grows in terms of iterations.

GB-CSCP is superior when observability is low

We define observability of an environment as the ratio between the maximal sensor coverage given the max sensor radius ϱ^{max} to the area of the environment. As visualized in Figures 3.2, 3.3, and 3.4, for low sensor counts the iterations required with increasing environment area is increasingly outperforming CSCP. This result is mainly attributed to the batching procedure near termination of the algorithm. In 'standard' CSCP with a low sensor count, it would take many iterations to polish the final solution with few sensors. However, batching with GB-CSCP enables a pseudo high sensor count solution in a single iteration, effectively reducing the number of required iterations to achieve convergence.

Batching reduces overall runtime

The runtime analysis in Table 3.2 shows that the GB-CSCP algorithm is capable of finding a solution in significantly less time than the CSCP approach. Figures 3.5, 3.6, and 3.7 show that with increasing sensor counts the GB-CSCP approach converges with increasingly less time. This speed advantage is attributed to the greedy pruning of the identified vertices. Additionally, while the batched optimization may be computationally more expensive in a single iteration, it typically converges much faster as previously stated than CSCP which reduces the entire duration runtime.

3.3 Demonstrative Example

To demonstrate the operation of GB-CSCP and depict it's benefits, consider the example field shown to the left in Figure 3.8 with a true optimal path-plan (white). Initially, the GB-CSCP algorithm behaves just like CSCP and it places the $N_s = 3$ available sensors within along the shortest path-plan region. The resulting estimated path-plan is shown by the green trace. The key difference is that the identified workspace vertices (green dots) are accounted for and displayed over the estimated threat field.

During the next several iterations, as shown in Figures 3.9 and 3.10, the algorithm performs sensor configuration over just the unidentified vertices. This enforces an exploration element which

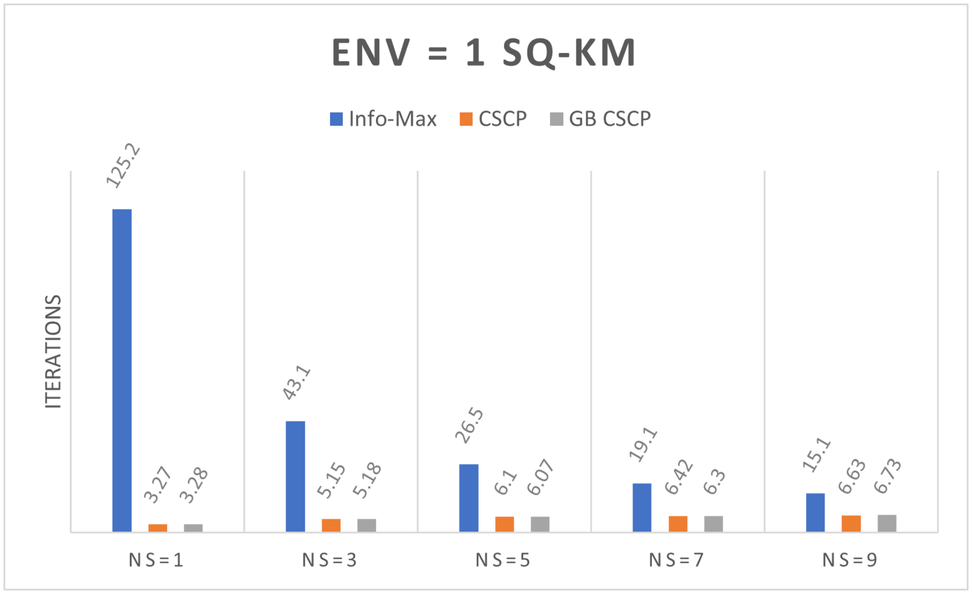


Figure 3.2: Iteration performance for environment information maximization, CSCP, and the Greedy Batched CSCP method for various numbers of available sensors, with an environment area of 1 square kilometer.

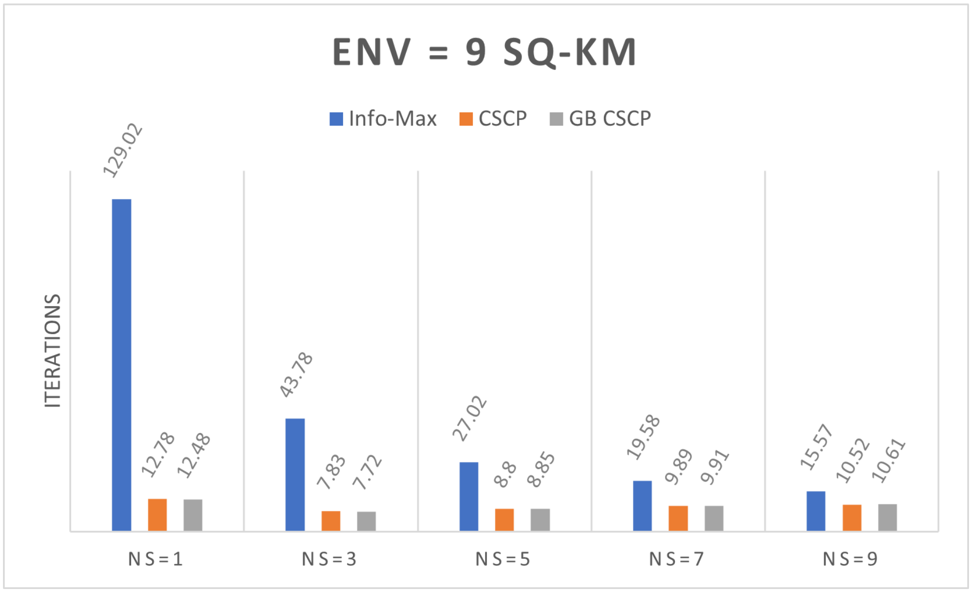


Figure 3.3: Iteration performance for environment information maximization, CSCP, and the Greedy Batched CSCP method for various numbers of available sensors, with an environment area of 9 square kilometers.

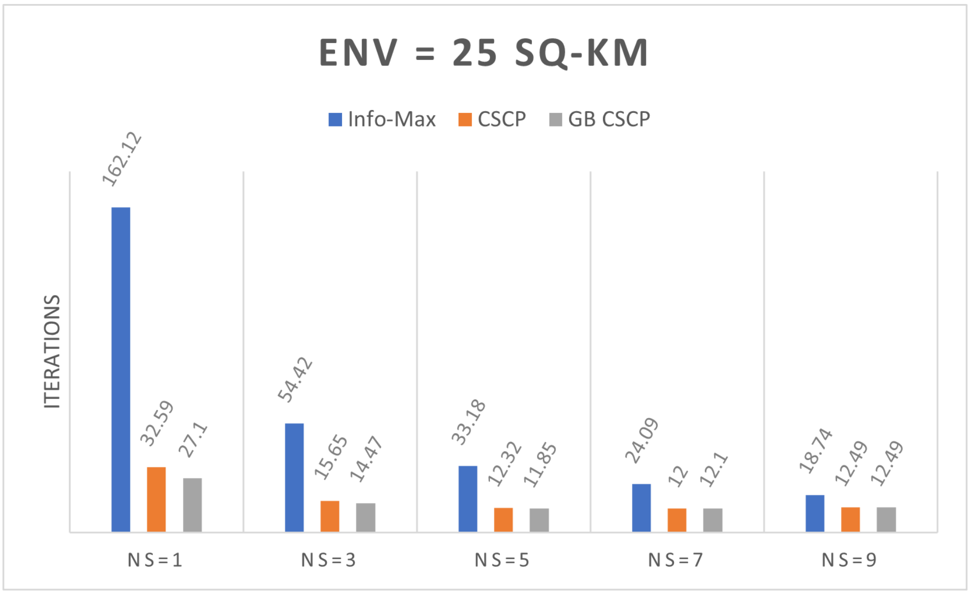


Figure 3.4: Iteration performance for environment information maximization, CSCP, and the Greedy Batched CSCP method for various numbers of available sensors, with an environment area of 25 square kilometers.

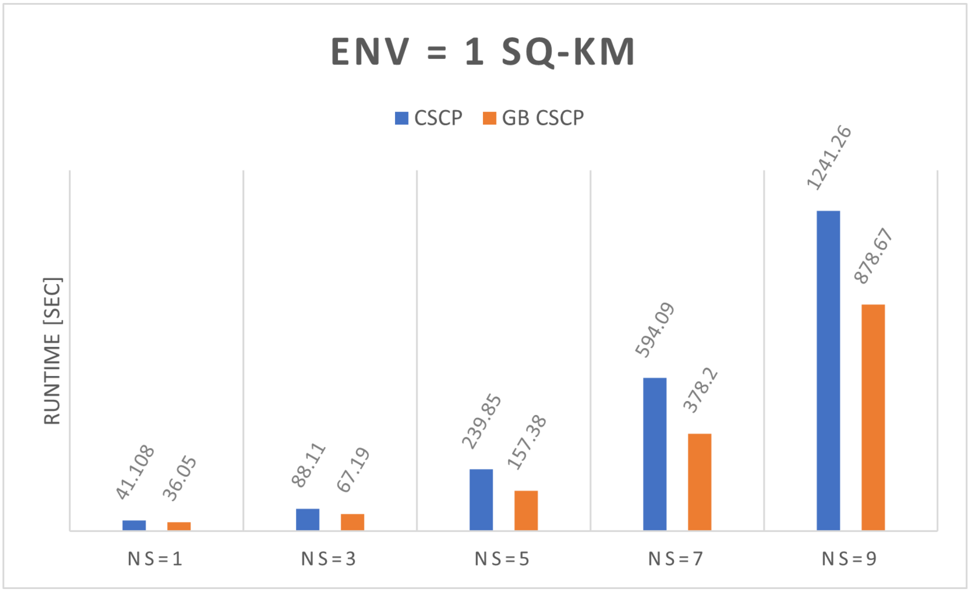


Figure 3.5: Runtime in seconds for CSCP, and the Greedy Batched CSCP method for various numbers of available sensors, with an environment area of 1 square kilometer.

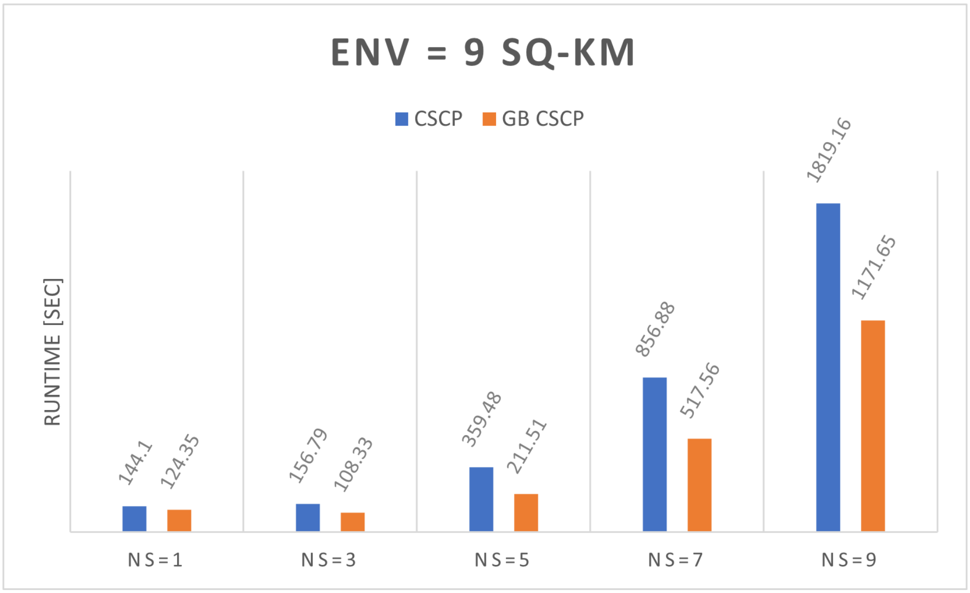


Figure 3.6: Runtime in seconds for CSCP, and the Greedy Batched CSCP method for various numbers of available sensors, with an environment area of 9 square kilometers.

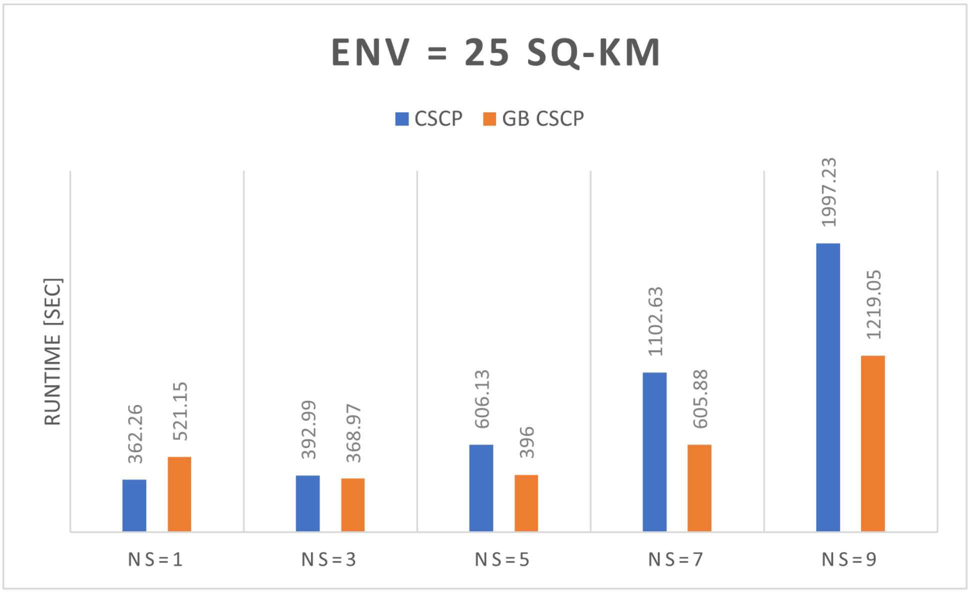


Figure 3.7: Runtime in seconds for CSCP, and the Greedy Batched CSCP method for various numbers of available sensors, with an environment area of 25 square kilometers.

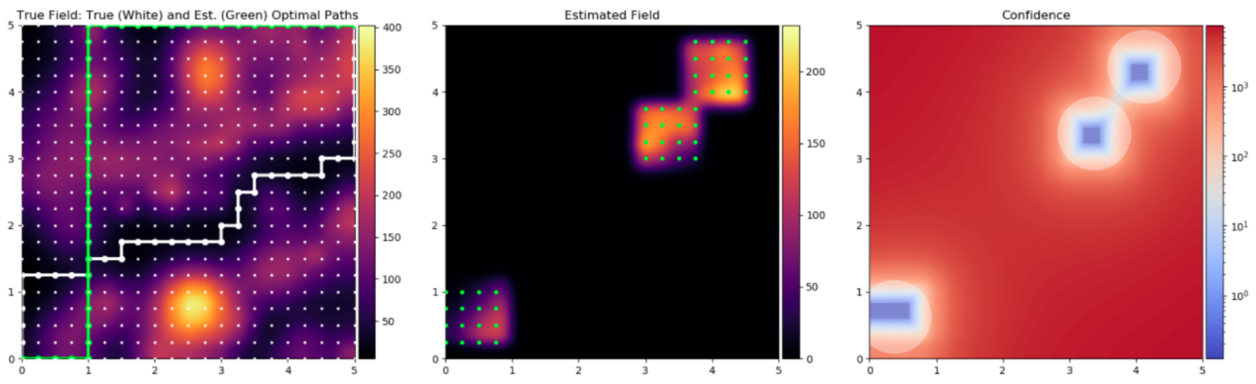


Figure 3.8: Initial GB-CSCP sensor configurations, estimated field, identified vertices, and new estimated optimal path-plan.

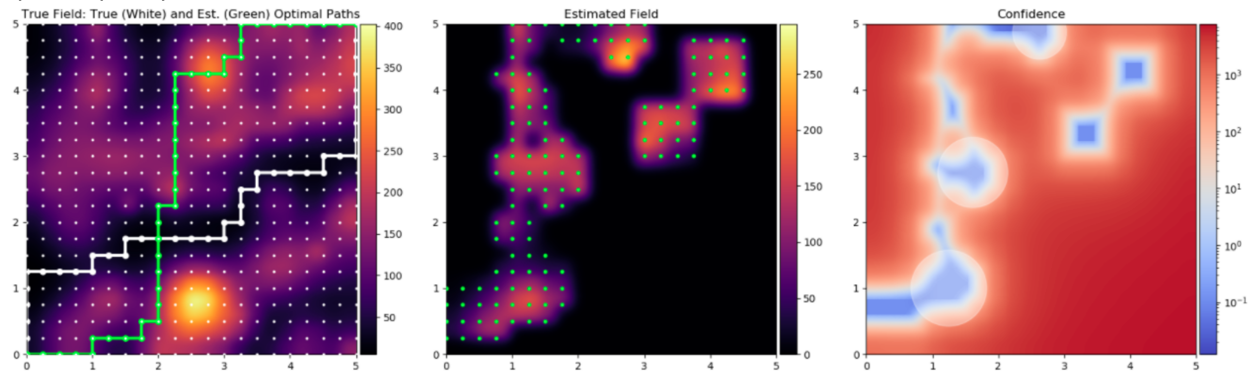


Figure 3.9: GB-CSCP sensor configurations, estimated field, identified vertices, and new estimated optimal path-plan at $\ell = 3$.

causes the sensor configuration to make a sweeping pattern across the environment.

Iteration $\ell = 12$ is the penultimate iteration as shown in Figure 3.11. We note that at this iteration, the algorithm has discovered the true optimal path-plan, but critically has not satisfied the confidence termination threshold and will not terminate. The CSCP algorithm, took four additional iterations to converge, ending with $\ell = 16$ iterations. However, the GB-CSCP algorithm converges below ε at $\ell = 13$ due to the batching property as previously described. Figure 3.12 shows the entire batched sensor configuration in the final iteration. In summary, we have shown how the GB-CSCP algorithm quickly explores the environment and then exploits the estimated path-plan for faster convergence in both iterations and computation time.

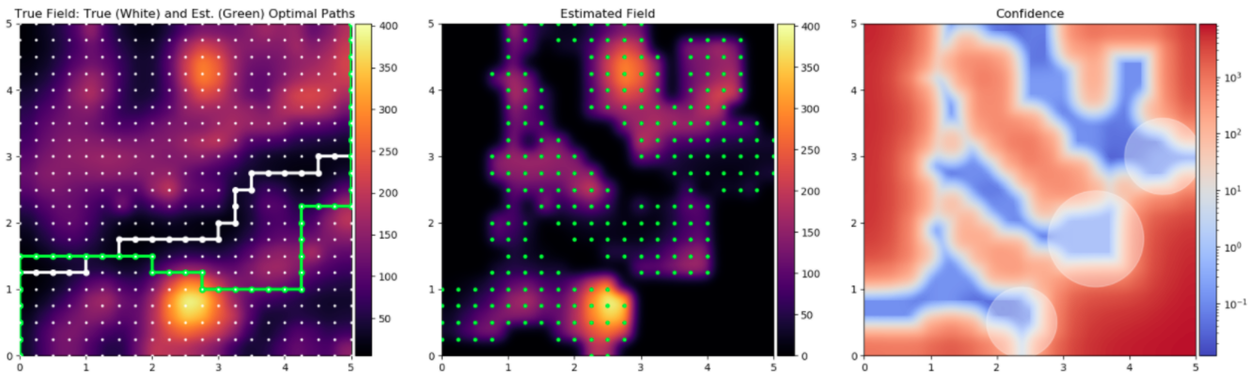


Figure 3.10: GB-CSCP sensor configurations, estimated field, identified vertices, and new estimated optimal path-plan at $\ell = 7$.

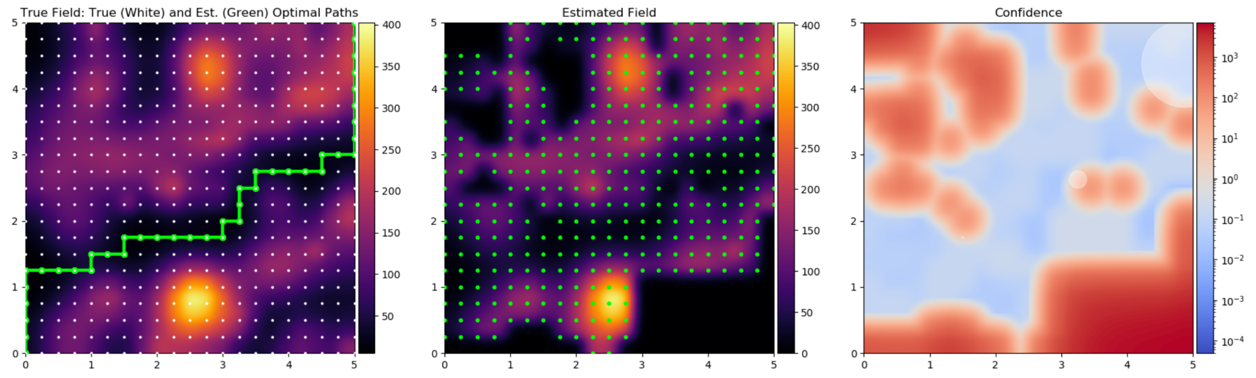


Figure 3.11: GB-CSCP sensor configurations, estimated field, identified vertices, and new estimated optimal path-plan at $\ell = 12$, the penultimate iteration.

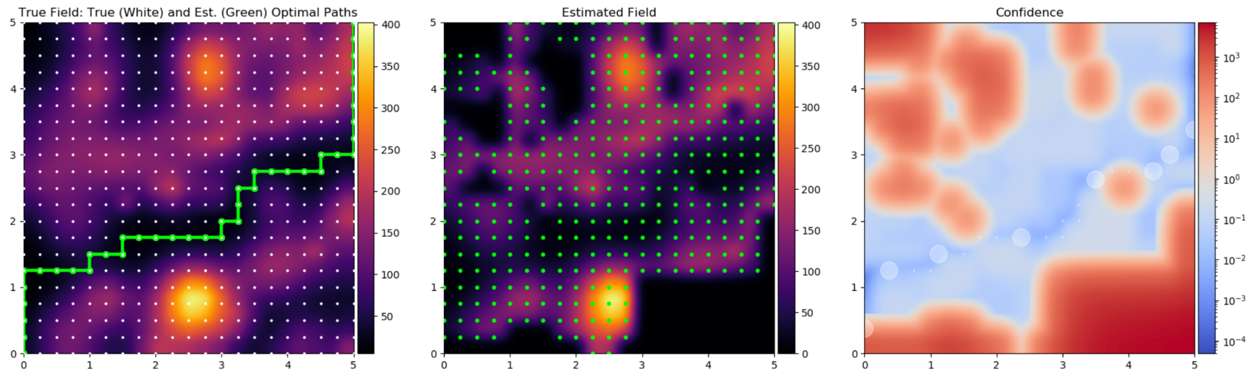


Figure 3.12: GB-CSCP sensor configurations, estimated field, identified vertices, and final estimated optimal path-plan at $\ell = 13$, the final iteration.

Chapter 4

Exploration Efficient CSCP with High Sensor Counts

4.1 Exploration Efficient CSCP

In situations requiring or possessing a high number of available sensors N_s, the gap in performance between the CSCP algorithm presented in Chapter 2 and information greedy approaches shrinks significantly. The nature of the CSCP algorithm is to place sensors directly along a current estimated optimal path plan. However, placing all of the available sensors along a single estimated optimal path may waste task-driven *exploration* opportunities when N_s is large. To make the CSCP methodology competitive with information greedy approaches for large N_s, we adopt an Exploration Efficient CSCP algorithm as detailed in Algorithm 4.1. The key differentiating factors are the ability to generate statistically feasible alternate path-plans from our statistical environment model. Secondly, we apply special weighting schemes to the region of interest vector $\boldsymbol{r}$ using these path-plans to perform task-driven exploration in addition to exploiting the current estimated optimal path-plan.

4.1.1 Generating Statistically Feasible Path-Plans

First, we are tasked with generating N_a statistically feasible alternate path-plans. The statistical model generated by the Gaussian Process regression provides an estimate of the threat field $\boldsymbol{f}$ and threat error covariance matrix $\boldsymbol{P}$. This output follows that of a multivariate normal distribution. As such, we may draw samples that we can find estimated optimal path-plans over. We may generate each i^{th} sample as:

$$\boldsymbol{f}_\ell^{(i)} = \boldsymbol{f}_\ell + \boldsymbol{A}\boldsymbol{g}_i \tag{4.1}$$

Exploration Efficient Coupled Sensor Configuration and Path-Planning

```
1:  Let $\ell := 0$, $\boldsymbol{f}_0 := \boldsymbol{0}, \boldsymbol{P}_0 := \chi \boldsymbol{I}$
2:  Solve for $\boldsymbol{\pi}_0^* := \arg\min \overline{\mathcal{J}}_0(\boldsymbol{\pi})$
3:  while $\mathrm{Var}_\ell(\boldsymbol{\pi}_\ell^*) > \varepsilon$ do
4:      Draw $N_a$ field estimates $\boldsymbol{f}_\ell^{(i)}$
5:      Solve for $\boldsymbol{a}_{\boldsymbol{\pi}}$ and $\boldsymbol{a}_i \; \forall \; i \in N_a$
6:      Compute $\boldsymbol{w}_{\boldsymbol{\pi}}$ as per Eqn. 4.3 or Eqn. 4.4
7:      Set $\boldsymbol{r} := \boldsymbol{w}_{\boldsymbol{\pi}}$
8:      Optimize Sensor Configuration $\mathcal{C}_\ell$
9:      Record new measurements $\boldsymbol{z}$
10:     Increment iteration counter $\ell := \ell + 1$
11:     Find GPR-based threat field estimate $\boldsymbol{f}_\ell$ and error covariance $\boldsymbol{P}_\ell$
12:     Find $\boldsymbol{\pi}_\ell^* := \arg\min \overline{\mathcal{J}}_\ell(\boldsymbol{\pi})$
```

Figure 4.1: Pseudocode for the Exploration Efficient CSCP algorithm to solve Problem 1.

The matrix $\boldsymbol{A}$ can be computed from the threat covariance matrix using cholesky decomposition as $\boldsymbol{A}\boldsymbol{A}^\intercal = \boldsymbol{P}_\ell$. The vector $\boldsymbol{g}_i$ is therefore the i^{th} sample vector's independent normal variates. By introducing generated threat field estimates using the statistical model, we may find N_a alternate potential path-plans by recomputing Dijkstra's algorithm. We may combine these alternate paths with the candidate optimal path and compute $\boldsymbol{r}$ as the region of interest incidence vector as described in the next section.

4.1.2 Region of Interest Weighting

Upon determining each i^{th} sampled threat field vector $\boldsymbol{f}^{(i)}$, we solve for $\boldsymbol{\pi}_{\boldsymbol{a}}^{(i)} := \arg\min \overline{\mathcal{J}}_\ell(\boldsymbol{\pi}) = \Delta p \; \boldsymbol{v}_{\boldsymbol{\pi}}^\intercal \; \boldsymbol{f}_\ell^{(i)}$, where $\boldsymbol{\pi}_{\boldsymbol{a}}^{(i)}$ is the i^{th} set of alternate estimated optimal path-plan vertices. We define by $\boldsymbol{a}_i$ the i^{th} alternate path incidence vector, where $\boldsymbol{a}_i[j] = 1$ if $j = \boldsymbol{\pi}_a^{(i)}[k]$ for $k \in [\lambda]\backslash 0$ and $\boldsymbol{a}_i[j] = 0$ otherwise. Similarly, we define the holistic set of alternate path plans and the current estimated optimal path plan as $\boldsymbol{\pi}_a$ (no superscript), which is determined by $\boldsymbol{\pi}_a := \boldsymbol{\pi}_\ell^* \cup \{\cup_{i=1}^{N_a} \boldsymbol{\pi}_a^{(i)}\}$. Likewise, we define it's incidence vector as $\boldsymbol{a}_{\boldsymbol{\pi}}$, where $\boldsymbol{a}_{\boldsymbol{\pi}}[i] = 1$ if $j = \boldsymbol{\pi}_a[j]$ for $j \in [\lambda]\backslash 0$ and $\boldsymbol{a}_{\boldsymbol{\pi}}[i] = 0$ otherwise. Next, we describe two region of interest weighting methods which utilize this information for enabling efficient task-driven exploration when N_s is large.

Breadth-First Exploration

The first weighted region of interest method we present exhibits sensor configuration exhibiting breadth-first search characteristics. To exhibit this behavior, we formulate a weighted vector as show in Equation 4.3.

$$\boldsymbol{\omega} := \frac{1}{N_a + 1}\left(\boldsymbol{v}_\pi + \sum_{i=1}^{N_a} \boldsymbol{a}_i\right) \tag{4.2}$$

$$\boldsymbol{w}_\pi := 2\boldsymbol{a}_\pi - \boldsymbol{\omega} \tag{4.3}$$

Equation 4.2 determines the frequency in which a vertex is part of an estimated optimal path plan. In Equation 4.3, we take the difference between twice the sampled path incidence vector and this frequency vector, resulting in a unit weight for vertices which are always a part of an optimal path-plan and extra weighting on vertices which are least frequent. The weighted region of interest vector can then be set to the region of interest vector $\boldsymbol{r} = \boldsymbol{w}_\pi$ and used during sensor configuration optimization. As a result, we enforce task-driven exploration at the 'fringe' of statistically feasible path-plans. The effect is the dispersal of many sensors to spatially different regions to narrow down feasible path-plan routes.

Entropic Exploration

The goal of the breadth-first weighted region of interest in Equation 4.3 was to effectively disperse large numbers of sensors to the 'inflection' points of possible path-plans. After narrowing down the statistically feasible path-plans the large numbers of sensors would operate similarly to the base algorithm presented in Chapter 2. However, while this weighted region of interest vector in Equation 4.3 helps the dispersal, a breadth-first search of statistically feasible path-plan regions may not be most beneficial in all applications. As such, we present an alternative which makes use of the vertex entropy given their frequency as embodied in the frequency vector $\boldsymbol{\omega}$.

Using the frequency vector $\boldsymbol{\omega}$ we can compute an entropic region of interest vector $\boldsymbol{e}_\pi$ as follows.

$$\boldsymbol{e}_\pi := -\boldsymbol{\omega}\log_2(\boldsymbol{\omega}) - (1 - \boldsymbol{\omega})\log_2(1 - \boldsymbol{\omega}) + \epsilon\boldsymbol{\omega} \tag{4.4}$$

This formulation calculated the entropy of each vertex by assuming each vertex was independent

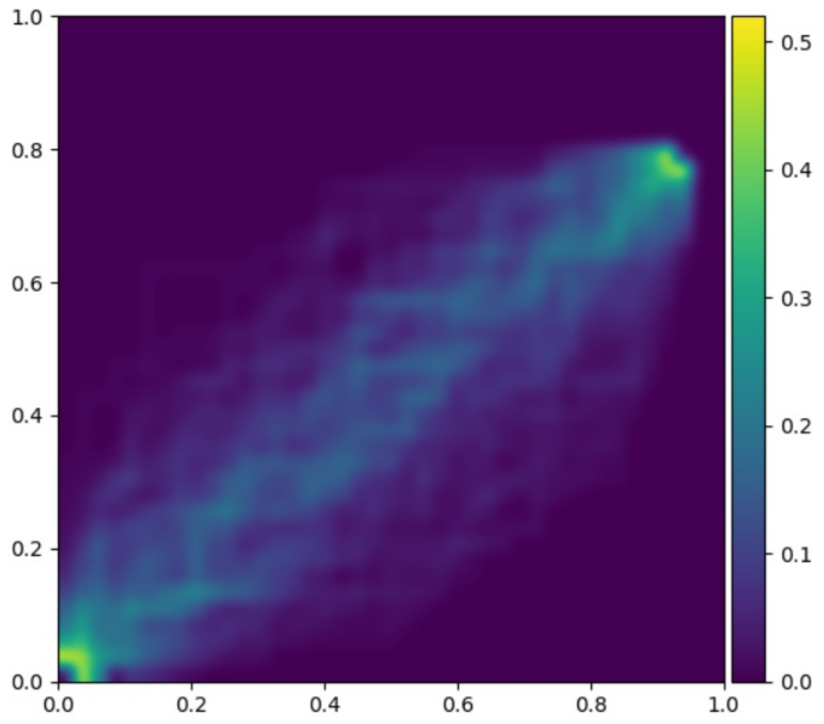


Figure 4.2: Alternate paths drawn from field estimate with the entropic exploration weighting applied.

and was modelled as a Bernoulli process representing it's membership of being in the optimal path-plan or not. Effectively, vertices which are not a part of the alternate plans or current estimated optimal path-plan, as well as vertices which are a part of every single path-plan, are given negligible weight characterized by a small scalar constant ϵ which prevents $\boldsymbol{\omega} = \mathbf{0}$. We can optimize the sensor configuration with the entropic region of interest vector by setting $\boldsymbol{r} := \boldsymbol{e}_{\boldsymbol{\pi}}$.

Figure 4.2 visualizes the entropic exploration weighting method on a workspace between the start and goal vertex using several sampled paths. Less frequent vertices are given a low information entropy weight whereas frequencies equal to approximately 50% achieve maximal entropy weighting which occurs near the start and goal. The mid-region has high frequency such that the frequencies are $> 50\%$ which causes the weighting to lessen.

4.2 Performance Comparison

The performance of the EE-CSCP method was compared against GB-CSCP, CSCP, and Info-Max on the same randomly generated environments. 100 randomly generated threat fields for each experiment were created of the form $c(\boldsymbol{x}) = \sum_{n=1}^{N_p} \theta_n \phi_n(\boldsymbol{x})$. The value N_p represents the number of radial basis functions ϕ_n that cover $\mathcal{E}$. The threat intensity θ_n is a coefficient that embodies the magnitude of each threat parameter. The number of threat parameters were fixed as $N_p = 50$ with a threat intensity of $\theta_n = 10$. We considered sensor network sizes of either N_s=3, 5, or 9 UAV's with minimum and maximum field of view radius constraints $\varrho_{\text{min}} = 0.01\text{km}$ and $\varrho_{\text{max}} = 0.5\text{km}$. The sensor noise is modeled as $\sigma_k^2 = \frac{1}{2}\log(1+\exp^{\pi\varrho_k^2}) - 0.1505$, which is monotonically increasing for $\varrho_k \geqslant 0$. The workspace resolution was varied between either N_g =225 or 400 and the environment axis size to be either 3km or 5km for a total area of $|\mathcal{E}| = 9\text{km}^2$ or $|\mathcal{E}| = 25\text{km}^2$, respectively. The number alternate paths for EE-CSCP were varied between options of N_a=25, 50 or 100. These we abbreviate as EE-25, EE-50, and EE-100 in the result plots, respectively. Finally, we set our termination threshold $\varepsilon = 0.01$.

The iterations required for convergence were the direct focus of this study as the EE-CSCP is concerned primarily with increasing the performance gap between the CSCP methodology and Info-Max. Recall that the standard CSCP taken along the estimated path-plan and GB-CSCP trend toward a shrinking performance in terms of iterations when the observability of the sensor network converges to the area of the environment. The following discussion provides commentary on the trends observed from the numerical results which are depicted in Figures 4.3, 4.4, 4.5, and 4.6.

EE-CSCP outperforms when the observability is high

Recall that we define the notion of 'observability' as the ratio between the area coverage of maximal sensor network field of view and the area of the entire workspace. Higher sensor counts therefore have higher observability than lower sensor counts, and when the workspace area increases while the sensor count remains the same, observability decreases and vice versa.

The drawback with both CSCP along the path-plan estimate and GB-CSCP are that the advantage in terms of iterations until convergence over Info-Max is lessened as observability becomes increasingly high. Observing the provided results, EE-CSCP outperforms the other CSCP even when the observability is increased, but the other CSCP methods do not perform nearly as well. As a result we can conclude that the EE-CSCP works as intended. It performs exploration over feasible path-plans and acts as a blend between the direct path-plan region of interest optimization

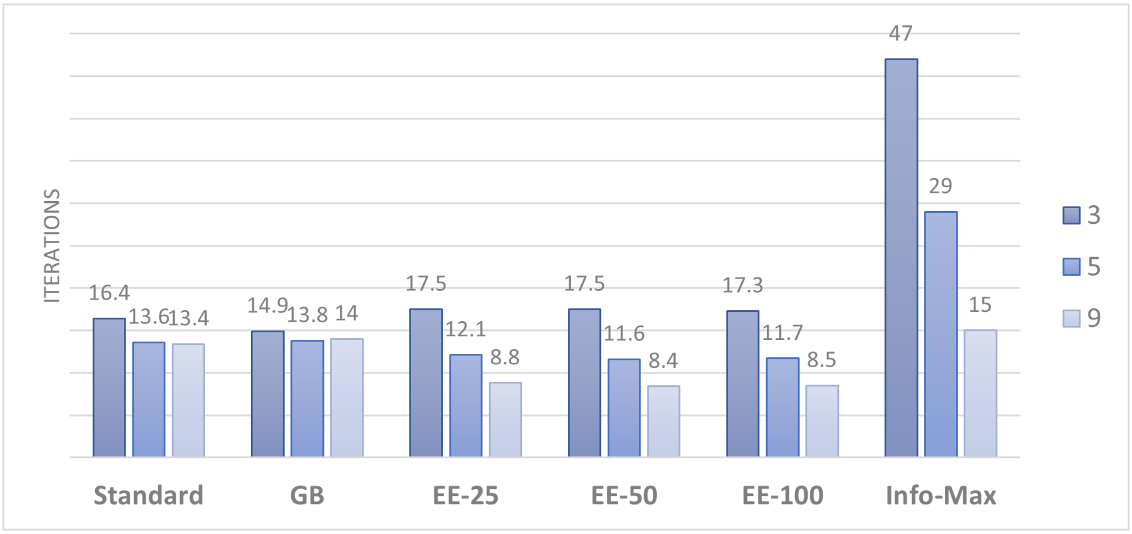


Figure 4.3: Iteration performance comparison of EE-CSCP and other methods with $|\mathcal{E}| = 9$ *and* $N_g = 225$.

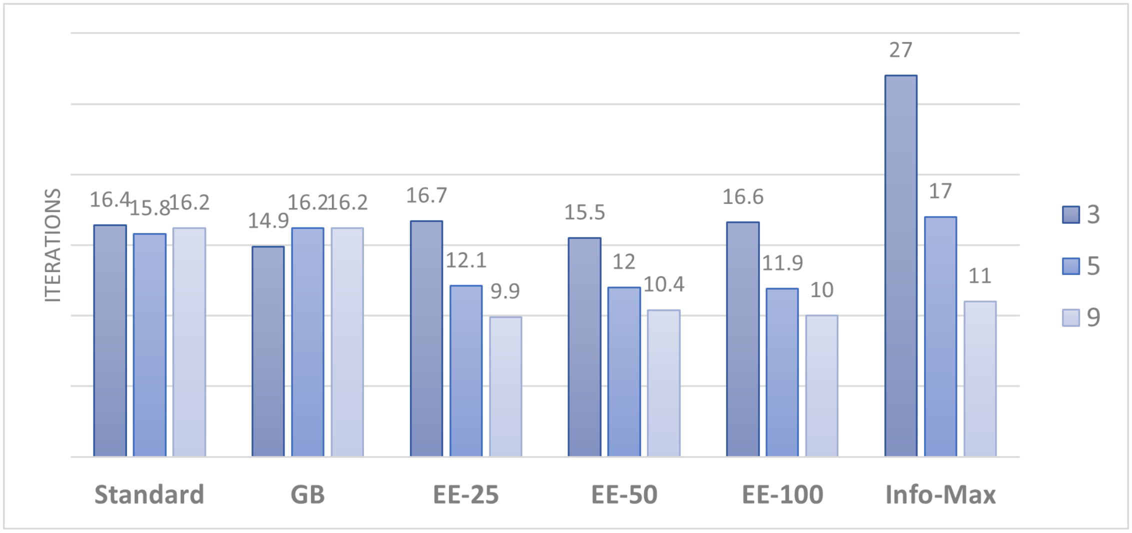


Figure 4.4: Iteration performance comparison of EE-CSCP and other methods with $|\mathcal{E}| = 9$ *and* $N_g = 400$.

and pure environment information maximization.

High alternate path counts N_a do not drive performance

Of important note is the effect of various alternate path counts on the iteration count performance. From the results it becomes evident that computing high numbers of alternate path-plans does not impact performance in any meaningful way. This is important because it ensures good iteration count performance without the need to spend computational resources generating many alternate path-plans for region of interest weighting.

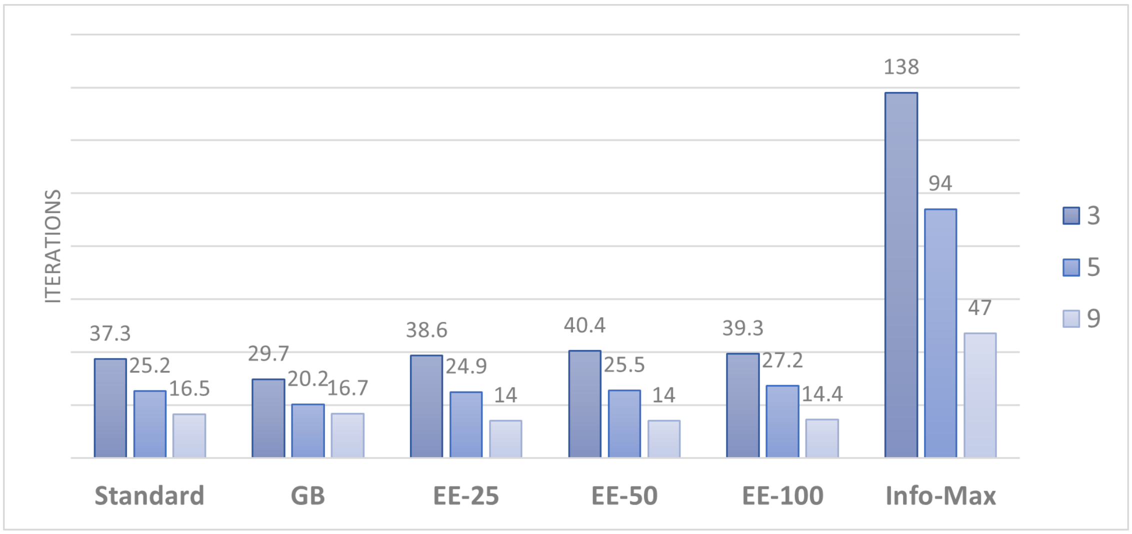


Figure 4.5: Iteration performance comparison of EE-CSCP and other methods with $|\mathcal{E}| = 25$ *and* $N_g = 225$.

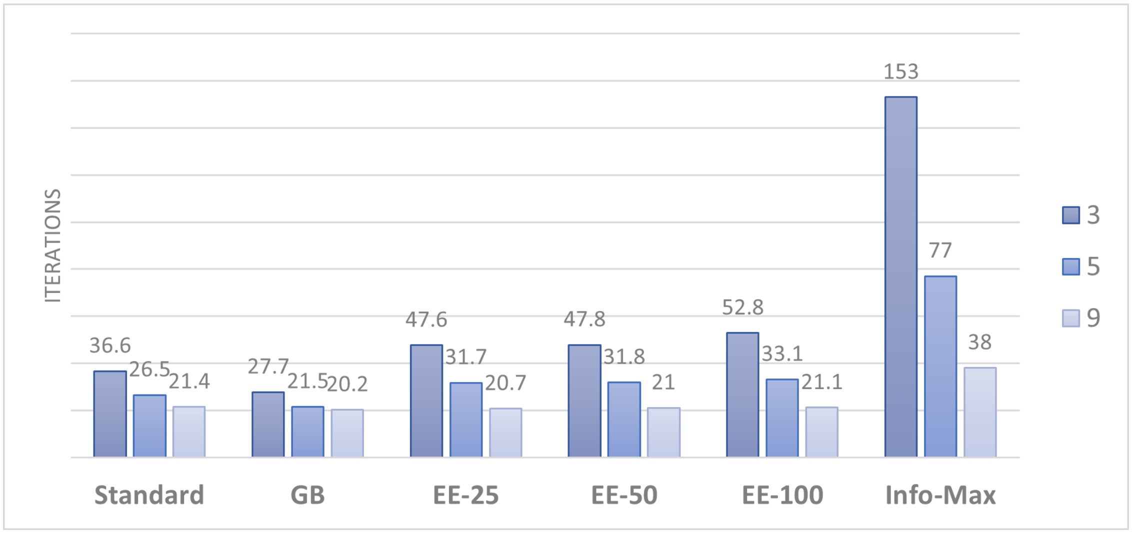


Figure 4.6: Iteration performance comparison of EE-CSCP and other methods with $|\mathcal{E}| = 25$ *and* $N_g = 400$.

Chapter 5

Qualitative Sensor Configuration for CSCP

5.1 Adaptive Cluster Analysis for CSCP

The methods described in Chapters 2, 3, and 4 all performed sensor configuration by directly optimizing the TDIG of Equation 2.3 using an approximation $\hat{\boldsymbol{P}}_{\ell+1}$ with various settings for the region of interest incidence vector $\boldsymbol{r}$. The joint optimization of N_s sensors which each have a position $\boldsymbol{s}_k$ and radial Fov parameter ϱ_k, which can lead to a high dimensional optimization problem rapidly as the number of dimensions N_d is defined as $N_d := 3N_{\text{s}}$ when considering just these sensor configuration parameters. If configuration options were to include arguments such as FoV yaw or pitch or similar, the dimensionality could rapidly grow. The overlapping element of sensor FoV leads to a non-convex optimization problem with many local optima requiring global optimization methods such as Bayesian Optimization or Differential Evolution. All of this is to say that the optimization of the sensor configuration in the CSCP algorithm can be computationally burdensome and the globally optimal solution $\mathcal{C}_\ell^*$ hard to find.

In scenarios in which a quick sensor configuration solution is required, but the resulting sensor configuration can be somewhat suboptimal, we present a method for qualitative sensor configuration which is capable of producing adaptive fast-approximations to the optimal sensor configuration. The fast-approximation is carried out either via a partition-based or density-based cluster analysis technique. The algorithm adaptively switches between these two modes, driven by the relationship between the region of interest uncertainty and the uncertainty of the entire environment.

Cluster analysis is an unsupervised learning procedure which is used in many disciplines for data grouping and class discovery without the need for data labels. A comprehensive overview of clustering algorithms and some applications are provided in (Xu and Wunsch, 2005). Clustering algorithms have been used to find sensor configurations, namely (Li et al., 2016) uses K-Means for

Adaptive Cluster Analysis for Sensor Configuration

```
 1: if $\bar{\boldsymbol{P}}_\ell[r,r] \geqslant \bar{\boldsymbol{P}}_\ell[i,i]$ then
 2:     Obtain $\mathcal{R}_*$ from W-KMeans$(\boldsymbol{p}_\mathcal{R}, \boldsymbol{P}_\ell[r,r])$
 3:     Find $\boldsymbol{s}_k$ and $\varrho_k$ per Eqn. 5.1 and 5.2
 4: else
 5:     Calculate $\boldsymbol{\omega} = (\boldsymbol{P}_\ell[r,r])^{-\frac{1}{2}}$
 6:     Find the transformed distance matrix $\boldsymbol{\mathcal{T}} = \boldsymbol{\omega}\boldsymbol{\omega}^\intercal \odot \boldsymbol{D}$
 7:     Obtain $\boldsymbol{\zeta}, \boldsymbol{\phi}_*, \mathcal{R}_*$ from HDBSCAN$(\boldsymbol{\mathcal{T}})$
 8:     $N_c = \min\{N_s, |\boldsymbol{\zeta}|\}$
 9:     Select clusters $\mathcal{R}_1, ..., \mathcal{R}_{N_c}$ from sort$(\boldsymbol{\zeta})$
10:     Find $\boldsymbol{s}_k$ and $\varrho_k$ as per Eqn. 5.3 and 5.2
11: for $k \in N_c$ do
12:     Constrain radius as $\varrho_k = \max\{\varrho^{\min}, \min\{\varrho^{\max}, \varrho_k\}\}$
13:     if $\max ||\boldsymbol{s}_k - \boldsymbol{p}_{\mathcal{R}_k[i]}||_2 < \varrho_k \quad \forall\, i \in |\mathcal{R}_k|$ then
14:         Translate $\boldsymbol{s}_k := \boldsymbol{p}_{\mathcal{R}_k[i]}$ for the $i^{th}$ closest point in $\mathcal{R}_k$
```

Figure 5.1: Pseudocode for sensor configuration with adaptive cluster analysis.

detecting degrees of freedom in frequency response functions for catching placement redundancies and in (Yoganathan et al., 2018) density-based clustering was used for optimal placement in office spaces. Evidential c-means clustering is utilized to find the minimum number of sensors for water leak monitoring (Sarrate et al., 2014).

We utilize clustering to provide multiple groupings of workspace vertices that the sensor network can be assigned to observe. This cluster analysis for sensor configuration (CLAN) strategy is outlined in Algorithm 5.1. CLAN utilizes an adaptive switching between *partition*-based and *density*-based clustering, which qualitatively correspond to *exploratory* and *exploitative* sensor configuration strategies, respectively. We show that the density-based clustering enables us to perform sensor configuration each iteration with a subset of available sensors, leading to savings in terms of the number of measurements required. In what follows, we describe the CLAN algorithm and show that the fast-approximation does not come at the expense of the overall CSCP algorithm performance.

5.1.1 CLAN Algorithm

The CLAN algorithm is a direct replacement for Line 4 of Algorithm 2.3, which calls for sensor configuration within the inner loop of the CSCP algorithm. CLAN makes use of the mean variance of the current iteration region of interest, denoted as $\bar{\boldsymbol{P}}_\ell[r,r]$ where 'r' are the indices along the diagonal corresponding to the set $\mathcal{R}$, and that of the entire environment, denoted as $\bar{\boldsymbol{P}}_\ell[i,i]$. The algorithm adapts to either perform exploratory or exploitative clustering given the relationship between $\bar{\boldsymbol{P}}_\ell[r,r]$ and $\bar{\boldsymbol{P}}_\ell[i,i]$. When the relationship exhibits $\bar{\boldsymbol{P}}_\ell[r,r] \geqslant \bar{\boldsymbol{P}}_\ell[i,i]$ we perform exploratory clustering, else exploitative. When the vertices along the path are not as variable as the environment, the inequality condition indicates we are near an optimal solution and therefore should observe only high variance groupings and reject the low variance vertices. In what follows, we describe the exploratory and exploitative clustering portions of the CLAN algorithm.

5.1.2 Exploratory Clustering

Qualitatively, exploratory clustering is concerned with spatial coverage rather than grouping spatially dense high variance regions. Such clustering can be accomplished using partition-based algorithms such as K-Means (Wang and Su, 2011). In this work, we utilize weighted K-Means++, which uses the variances of each vertex as weights and an initialization strategy as described in (Arthur and Vassilvitskii, 2007). Other partition-based methods such as K-Medoids (Kaufman and Rousseeuw, 1990) may be used, which is a robust clustering algorithm which is less sensitive to outliers. The data points are defined as the coordinates in our region of interest set $\boldsymbol{p}_\mathcal{R}$ and for the weighted data points used during clustering, we utilize $\boldsymbol{P}_\ell[r,r]$. In K-Means we need to determine the number of clusters N_c to partition the data into. For our application, the typical cross-validation steps are not required. We simply need to set the number of clusters $N_c = N_s$, effectively enforcing that each sensor belongs to a resulting cluster.

The weighted K-Means algorithm returns N_c partitioned sets of the region of interest set as $\mathcal{R}_* := \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_{N_c}\}$. Using these partitioned data sets, we can compute the average coordinate and radius value. As we have $N_s = N_c$ we can say that each i^{th} cluster corresponds to directly solving for each k^{th} sensor. For each i^{th} set, we determine the sensor position $\boldsymbol{s}_k$ of the set $\mathcal{R}_i$ as:

$$\boldsymbol{s}_k = \frac{1}{|\mathcal{R}_i|} \sum_{j=0}^{|\mathcal{R}_i|} \boldsymbol{p}_{\mathcal{R}_i[j]} \tag{5.1}$$

The radius value of the i^{th} region of interest partition ϱ_k is then formulated as:

$$\varrho_k := \max ||\boldsymbol{s}_k - \boldsymbol{p}_{\mathcal{R}_i[j]}||_2, \quad \forall\, j \in |\mathcal{R}_i| \tag{5.2}$$

5.1.3 Exploitative Clustering

When the average vertex variance along path vertices is less than the average vertex variance of the entire environment, we perform exploitative clustering. Exploitative clustering finds clusters of high density, or tight groupings of high variance vertices. We utilize Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) (Campello et al., 2013), which finds dense groupings and rejects outliers. HDBSCAN determines the relative density of data points using a notion of core density. However, we override this by computing a transformed distance matrix $\boldsymbol{\mathcal{T}} := \boldsymbol{\omega}\boldsymbol{\omega}^\intercal \odot \boldsymbol{D}$, where we compute the element-wise product of $\boldsymbol{\omega} := \boldsymbol{P}_\ell[r,r]^{-\frac{1}{2}}$ and a region of interest distance matrix $\boldsymbol{D}$. The operator $\odot$ is the element-wise product. The distance matrix $\boldsymbol{D}$ is defined as the pairwise distance between points $\boldsymbol{D}[i,j] := ||\boldsymbol{p}_{\mathcal{R}_i} - \boldsymbol{p}_{\mathcal{R}_j}||_2\ \forall\, i,j \in |\mathcal{R}|$. Analyzing this construction, notice that large values of $\boldsymbol{P}_\ell[r,r]$ correspond to small values of $\boldsymbol{\omega}$ and vice versa. The effect, which is desired, is that high variance vertices have a spatially attractive force and low variance vertices have a spatially repellent force.

We then perform HDBSCAN with a minimum cluster size of 2, meaning we can have a cluster with minimally only two vertices, and extract clusters pertaining to leafs on the condensed cluster tree. HDBSCAN returns a vector of the stability of each cluster $\boldsymbol{\zeta}$ along with a set of class probability vectors for each vertex $\boldsymbol{\phi}$ in each cluster set denoted as $\boldsymbol{\phi}_* = \{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_{N_c}\}$, in addition to the partition of the region of interest $\mathcal{R}_* := \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_{N_c}\}$. We define the resulting number of clusters N_c differently for HDBSCAN. To determine the number of clusters, we take the minimum between the number of available sensors and the length of the cluster stability vector $N_c = \min\{N_\text{s}, |\boldsymbol{\zeta}|\}$. Once the number of clusters is determined we set the number of configurable sensors to be the number of clusters. Of importance here is that this allows us to configure less sensors than are physically available. In order to determine which clusters from the partitioned region of interest to select, we sort the cluster stability vector and choose the top N_c clusters associated with their index. We also note that in the event HDBSCAN classifies all elements in $\mathcal{R}$ as a noise cluster, we treat this as a valid single cluster.

After obtaining the clusters by exploitative clustering, we can find the sensor position as the weighted average of points in each cluster. For $k \in N_c$ we find $\boldsymbol{s}_k$ as:

$$\boldsymbol{s}_k := \left(\sum_{i=0}^{|\phi_k|} \phi_k[i] \right)^{-1} \left(\sum_{i=0}^{|\phi_k|} \phi_k[i] \boldsymbol{p}_{\mathcal{R}_k[i]} \right) \tag{5.3}$$

We note that we may find the k^{th} sensor radius ϱ_k using Equation 5.2.

5.1.4 Cluster Check and Polishing

For both exploratory and exploitative clustering, we need to account for constraints on the sensor radius. This is accomplished by taking the max-min of the constraints with the output of Equation 5.2 as $\varrho_k = \max\{\varrho^{\min}, \min\{\varrho^{\max}, \varrho_k\}\}$. Due to this update, we need to perform a cluster check to ensure that the sensor FoV covers vertices in its region of interest partition. We need to perform a correction if the following condition holds true:

$$\max ||\boldsymbol{s}_k - \boldsymbol{p}_{\mathcal{R}_k[i]}||_2 < \varrho_k \quad \forall\, i \in |\mathcal{R}_k| \tag{5.4}$$

If Equation 5.4 holds true, we perform a correction by shifting the k^{th} sensor position $\boldsymbol{s}_k$ to the closest data point, $\boldsymbol{s}_k := \boldsymbol{p}_{\mathcal{R}_k[i]}$ for the i^{th} closest point in $\mathcal{R}_k$ to the pre-correction $\boldsymbol{s}_k$.

An additional benefit of clustering is the ability to use it as a dimensionality reduction technique for the direct optimization of the TDIG metric. By fixing the cluster location parameters, we may optionally *polish*, the solution by optimizing the low-dimensional optimization problem which only considers the radius parameter for each sensor.

5.2 Performance Comparison

The CSCP, CLAN, EE-CSCP, Info-Max, and Info-Max with the CLAN methodology were studied numerically. Additionally, polishing as previously described was applied to each method and for simplicity methods where this is applied have additional notation '-P'. 100 random trials on environment areas of $9km^2$, $25km^2$, $49km^2$ with workspace resolutions N_g of size 11^2, 21^2, 31^2, 41^2, 51^2 and sensor counts of 1, 3, 5, 7 or 9. Maximum and minimum sensor radius constraints were $0.1km$ and $1km$, respectively. The sensor noise is modeled as $\sigma_k^2 = \frac{1}{2}\log(1+\exp^{\pi\varrho_k^2}) - 0.1505$, which is monotonically increasing for $\varrho_k \geqslant 0$. Each trial was performed with randomly generated threat fields $c(\boldsymbol{x}) = \sum_{n=1}^{N_p} \theta_n \phi_n(\boldsymbol{x})$, where θ_n are randomly chosen coefficients, and ϕ_n are radial basis functions that cover $\mathcal{E}$. The number of bases N_p was fixed at $N_p = 50$. Figures 5.2 and 5.3 show the average iterations required to converge and overall runtime for each method for various sensor counts. We make the following comments about the obtained results.

CLAN significantly reduces runtime with slight performance reduction over direct optimization of the TDIG metric

The main motivation for the CLAN method for sensor configuration optimization was to provide fast-approximations for the sensor location and radius values. The results yielded a 71% runtime reduction over CSCP at the expense of an average increase of 2.81 iterations for all studied $N_s > 1$. We exclude $N_s = 1$ in these results as it is an outlier of poor performance most likely derived from the fact that exploratory clustering for $N_s = 1$ is the entire region of interest. We note that CLAN does not scale well in terms of average iteration performance when exploration is introduced with EE-CSCP or the Info-Max approaches.

Exploitative Clustering reduces required observations

A less obvious feature of the exploitative clustering stage is the ability to place $N_s = N_c$ sensors which may be less than the total number of available sensors. Again, excluding $N_s = 1$ cases, the CLAN algorithm averages 1.182 fewer sensor FoV configurations per iteration. Likewise, the polished CLAN algorithm achieves 1.65 fewer sensor FoV configurations per iteration on average. Thus, the CLAN strategy not only increases the speed for sensor configuration optimization, but also reduces the number of configured sensors and as a result the number of required observations for convergence.

Polishing CLAN improves performance while maintaining partial runtime reduction

We presented the idea of fixing the sensor configuration locations found by exploratory or exploitative clustering, but further optimizing on just the sensor radius parameters. The dimensionality reduction technique maintained an average runtime reduction of approximately 20% at the expense of an extra 3.4 iterations on average. We note that the large disparity is biased at low sensor counts. Removing the condition where $N_s = 1$, the CLAN strategy with polishing only takes and extra 0.6 iterations on average and increased runtime reduction of 31%.

CLAN and Polished CLAN do not scale well to EE-CSCP or Info-Max

The limitation with the CLAN approach with or without polishing is a poor performance when the scope of the region of interest includes more than the current estimated optimal path. Both EE-CSCP and Info-Max, when CLAN is applied, nearly double in average iterations to converge despite their runtime improvements. However, there is little to no advantage of using CLAN when information-greedy approaches such as Info-Max are applied.

5.3 Applied Comparison

Each of the methods studied in the numerical results and discussion were assessed over 25 iterations on an example threat field. The path variance was recorder at each iteration for each method using $N_s = 3$ on a $9km^2$ workspace with $N_g = 21$. No termination threshold was enforced in order to assess the progression over the finite 25 iterations. The path variance for each approach is shown in Figure 5.4.

The key results were that while the CSCP algorithm took 181 seconds to finish 25 iterations,

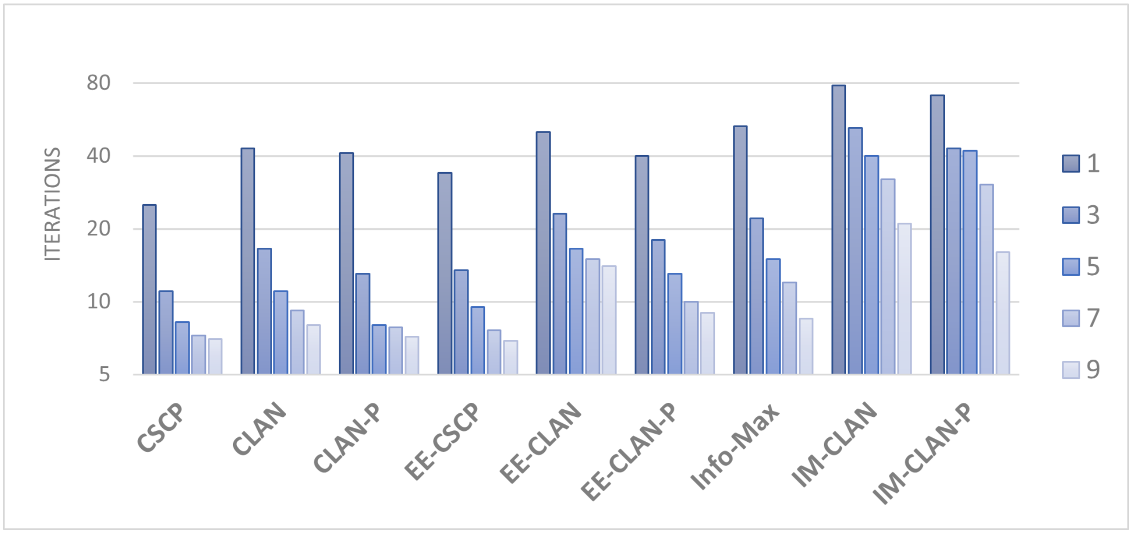


Figure 5.2: Comparison of iterations required for CSCP, EE-CSCP, and Info-MAX with CLAN and polished CLAN with various sensor counts.

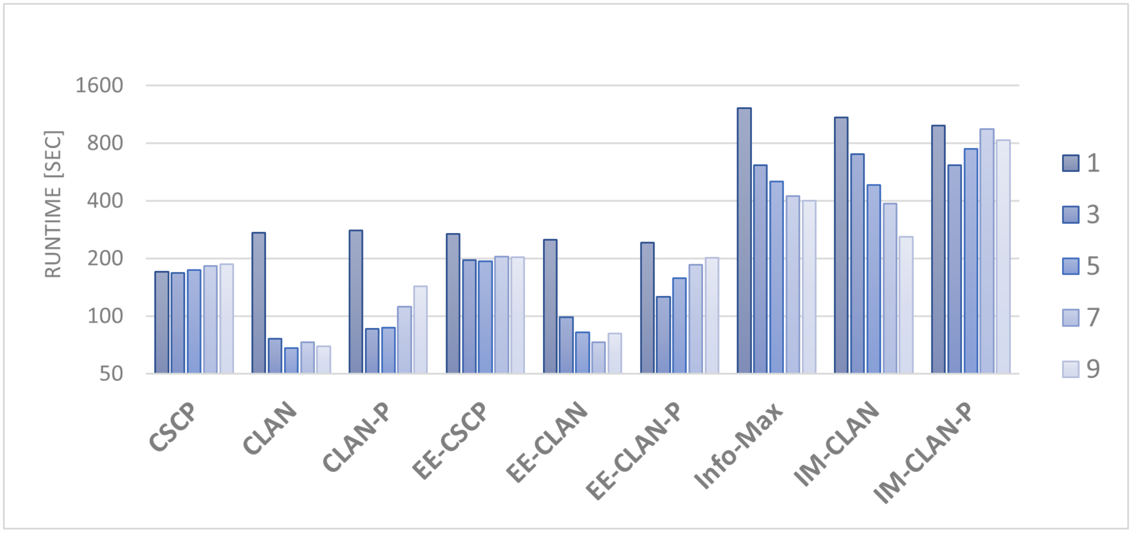


Figure 5.3: Comparison of average runtime for CSCP, EE-CSCP, and Info-MAX with CLAN and polished CLAN with various sensor counts.

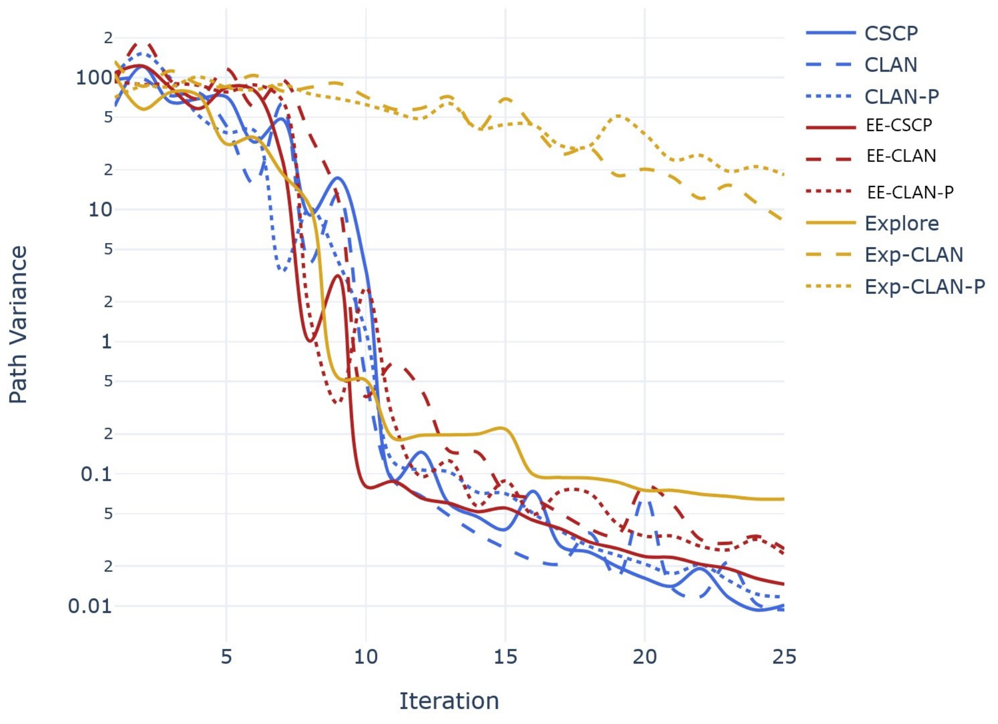


Figure 5.4: Path variance for each candidate approach at each iteration up to $\ell = 25$

the CLAN method only took 67 seconds. Polishing the CLAN results took 121 seconds to complete 25 iterations in this example. Note that in Figure 5.4, the CSCP methods also converge much faster to stricter path variance values than Info-Max (Explore). We also note that in this figure the EE-CSCP is the first to drop below 0.1 with only 10 iterations. Of significance is that the CLAN approaches keep pace with the direct sensor configuration optimization approaches. Additionally, the CLAN method is shown perform poorly when applied to an Info-Max strategy.

Figures 5.5, 5.6, and 5.7 depict the initial and 10^{th} iteration threat field estimates and estimated optimal path-plans for the various methods from the experiment. Qualitatively, the CLAN approaches applied to CSCP and EE-CSCP map more of the environment due to exploration during clustering. The polished variation is more conservative and more closely matches CSCP. When CLAN is applied to Info-Max the quality of the field estimate is reduced and the estimated path-plans are poor, reinforcing the poor performance of CLAN for Info-Max in the numerical study.

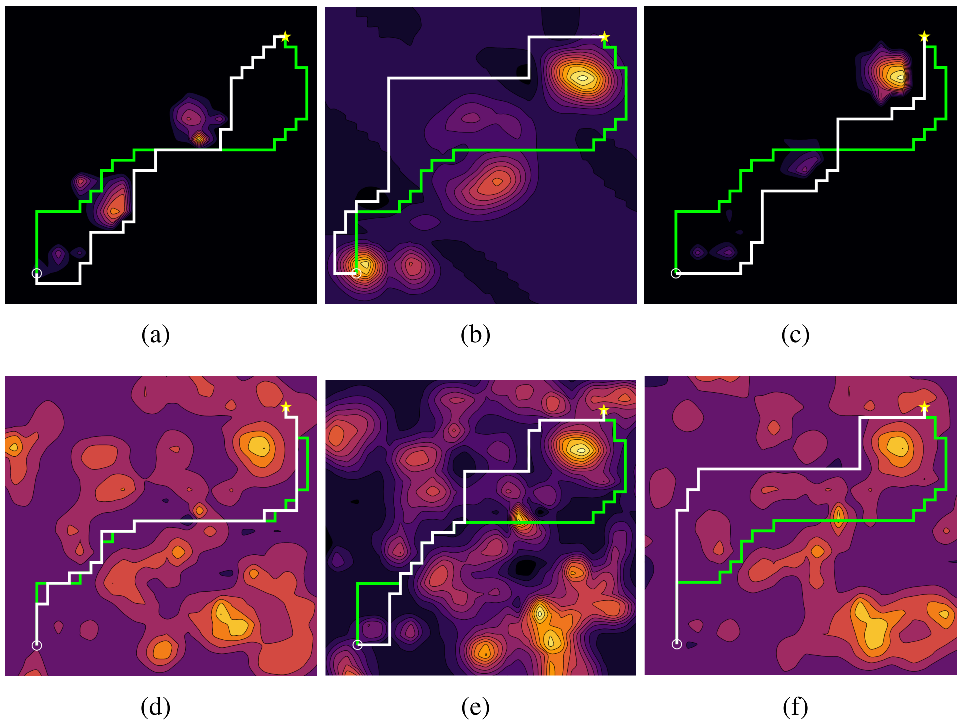


Figure 5.5: Example field estimate and estimated path-plan (white) and true path-plan (green) for the initial CSCP (a), initial CLAN (b), initial CLAN-P (c), 10th CSCP iteration (d), 10th CLAN iteration (e), and 10th CLAN-P iteration (f).

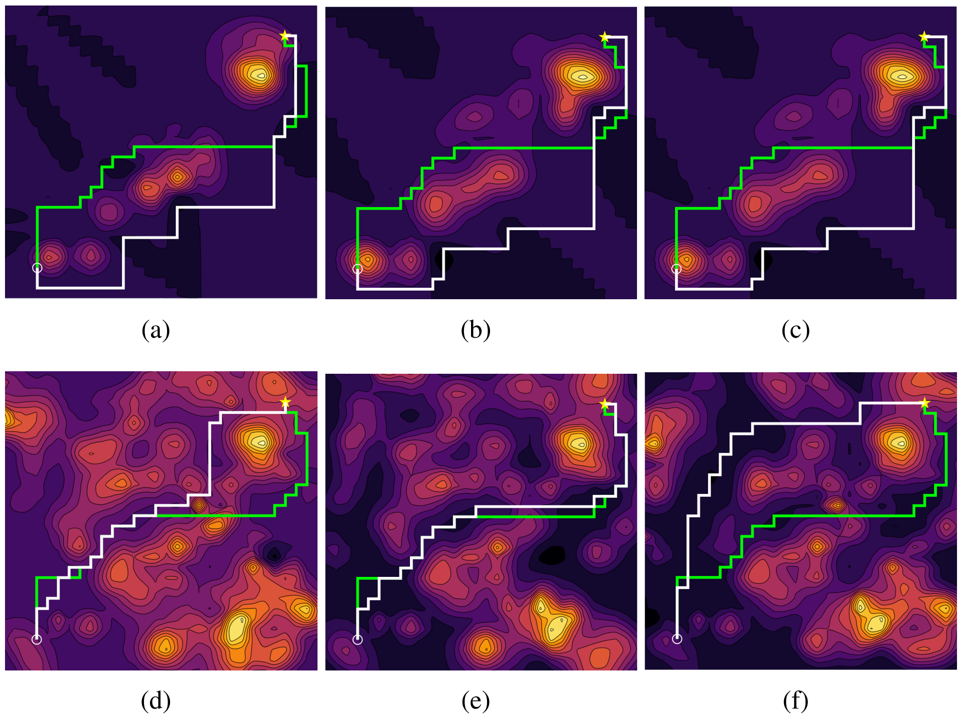
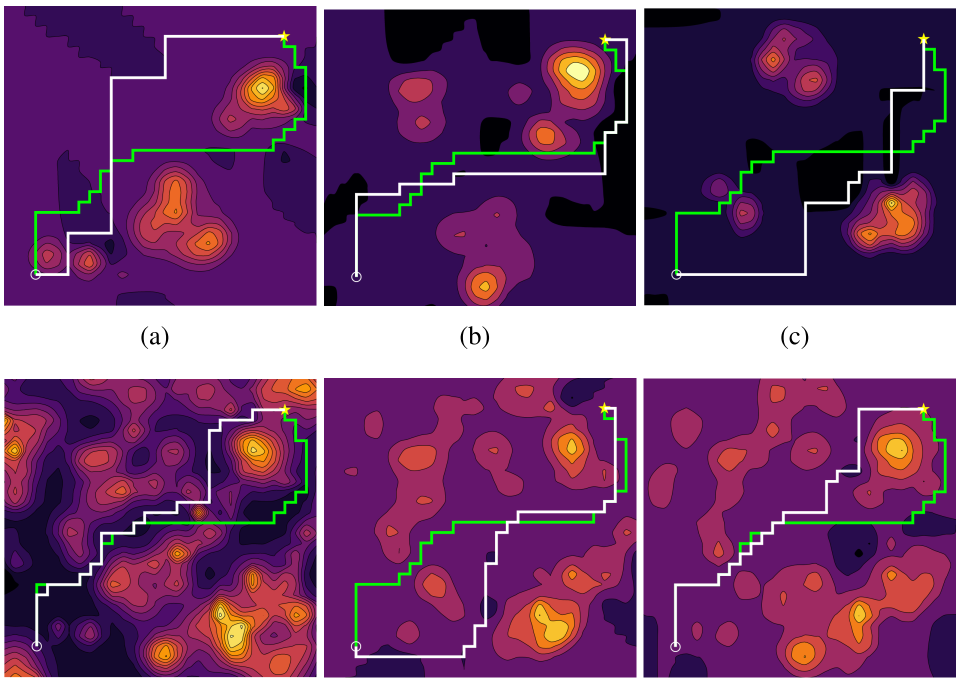


Figure 5.6: Example field estimate and estimated path-plan (white) and true path-plan (green) for the initial EE-CSCP (a), initial EE-CLAN (b), initial EE-CLAN-P (c), 10th EE-CSCP iteration (d), 10th EE-CLAN iteration (e), and 10th EE-CLAN-P iteration (f).



(a) (b) (c)

(d) (e) (f)

Figure 5.7: Example field estimate and estimated path-plan (white) and true path-plan (green) for the initial Info-Max (a), initial IM-CLAN (b), initial IM-CLAN-P, 10^{th} Info-Max iteration (d), 10^{th} IM-CLAN iteration (e), and 10^{th} IM-CLAN-P iteration (f).

Chapter 6

Self-Adaptive Mutual Information for CSCP

6.1 Problem Motivation

In the previous chapters, we detailed the use of the task-driven information gain (TDIG) metric for quantitatively assessing the performance of sensor configuration. In Chapter 2 we described that the sensor configuration with TDIG required an approximation to the posterior threat error covariance matrix, either requiring potentially overestimating performance when fixing correlations or potentially underestimating performance when assuming independence amongst vertices. While the TDIG provides a useful metric *post* sensor configuration and model fitting, it makes for a hard optimization problem. Additionally, the formulation is non-convex nor submodular, making the search for a near global optimal solution require joint optimization with a black box global optimizer. Given the problem domain has increasingly high dimensionality with the addition of each sensor, the joint optimization does not scale favorably. In Chapter 5 we showed a way to reduce the joint optimization computational burden with a qualitative clustering approach, yet this came with the trade-off of some performance inefficiencies.

The second critique of sensor configuration optimization with the TDIG metric is it has no means to account for situations in which we should explore non-task-driven regions of interest and when to exploit these regions of interest. In Chapter 3 we did exploration, but it was still confined to our notion of the region of interest. Similarly, in Chapter 4 we demonstrated the ability to perform exploration along statistically feasible regions of interest, but still neglected the domain outside of this region.

In what follows, we present a surrogate optimization function for TDIG which enables *sequential* optimization and has a self-adaptive property which allows for a balance between exploring the environment and exploiting our definition of a region of interest.

6.2 Surrogate Optimization Function

In previous chapters, we solved line 4 of Algorithm 2.3, sensor configuration, by optimizing the TDIG metric along a variety of notions for the task-driven region of interest. The optimization of this metric involved approximating the metric by either fixing the correlations between workspace vertices or assuming independence. To counter the drawbacks of optimizing TDIG directly, we present a surrogate optimization function, which we call *self-adaptive mutual information (SAMI).* SAMI is calculated at each iteration $\ell \in \mathbb{N}$. For notation simplicity we denote $\boldsymbol{P}_\ell[\mathcal{R},\mathcal{R}] := \boldsymbol{P}[i,j] \ \forall \ i,j \in \mathcal{R}$ and by $\boldsymbol{P}[\mathcal{R},j] := \boldsymbol{P}[i,j] \ \forall \ i \in \mathcal{R}$ and for any $j \in V$. Likewise, we say the region of interest set excluding a point $i \in V$ is written as $\mathcal{R}_{\setminus i}$. We break up the presentation of SAMI into two parts, the reward term and the penalty term.

Each iteration $\ell = 0, 1, \dots, L$ we perform Gaussian Process Regression and perform inference for threat field estimate $\boldsymbol{f}_\ell$ and threat error covariance matrix $\boldsymbol{P}_\ell$ which follows that of a multivariate normal distribution. From the threat error covariance matrix we are able to calculate the entropy of a vertex $i \in V$ as:

$$h(i) := \frac{1}{2}\ln(2\pi e \boldsymbol{P}_\ell[i,i]) \tag{6.1}$$

We rely on the conditional entropy of some point i given the set $\mathcal{R}$ which involves matrix inversion. To ease the computational burden, we note that the computation of conditional entropy $h(i|\mathcal{R}_{\setminus i})$ can partially be computed as a one-time batch operation and partially computed in parallel for efficiency. First, for any $i \notin \mathcal{R}$, we may perform the following vectorized conditional entropy $\boldsymbol{h}(\cdot|\mathcal{R}_{\setminus i})$ calculation which is a one-time batch operation:

$$\boldsymbol{h}(\cdot|\mathcal{R}_{\setminus i}) = \frac{1}{2}\ln((2\pi e) diag(\boldsymbol{P}_\ell - \boldsymbol{P}_\ell[\cdot,\mathcal{R}]\boldsymbol{P}_\ell[\mathcal{R},\mathcal{R}]^{-1}\boldsymbol{P}_\ell[\mathcal{R},\cdot])) \tag{6.2}$$

For the case where $i \in \mathcal{R}$, we can compute the following conditional entropy equation for any $i \in V$ in parallel batches:

$$h(i|\mathcal{R}_{\setminus i}) = \frac{1}{2}\ln((2\pi e)(\boldsymbol{P}_\ell[i,i] - \boldsymbol{P}_\ell[\mathcal{R}_{\setminus i},i]^\intercal \boldsymbol{P}_\ell[\mathcal{R}_{\setminus i},\mathcal{R}_{\setminus i}]^{-1}\boldsymbol{P}_\ell[\mathcal{R}_{\setminus i},i])) \tag{6.3}$$

The mutual information I between the $\mathcal{R}$ and $i \in V$ is then calculated as:

$$I(\mathcal{R}_{\setminus i}; i) := h(i) - h(i|\mathcal{R}_{\setminus i}) \tag{6.4}$$

We can then define the reward function of observing any point $i \in V$ as a mixture of mutual information gain between the region of interest and region of interest complement $\mathcal{R}^{\mathrm{c}} := V \setminus \mathcal{R}$. Note that we may compute the necessary region of interest complement conditional entropy as per Equations 8.1 and 8.2. The reward function for any point $i \in V$ is defined as:

$$\gamma(i) := (1 - \alpha) I(\mathcal{R}_{\setminus i}; i) + \alpha I(\mathcal{R}^{c}_{\setminus i}; i) \tag{6.5}$$

The variable α is an *adaptation parameter* which balances exploration and exploitation through proportional weighting of the mutual information gain and is bounded $0 \leqslant \alpha \leqslant 1$ where $\alpha \in \mathbb{R}$. The adaptation parameter is formulated as the relationship between the average mutual information between the workspace vertices and the ROI $\bar{I}(\mathcal{R}_{\setminus i}; i)$ and the average mutual information between the workspace vertices and the ROI complement $\bar{I}(\mathcal{R}^{c}_{\setminus i}; i)$:

$$\alpha := \bar{I}(\mathcal{R}_{\setminus i}; i) / \big(\bar{I}(\mathcal{R}_{\setminus i}; i) + \bar{I}(\mathcal{R}^{c}_{\setminus i}; i)\big) \tag{6.6}$$

Informally, α compares the estimation error within the current ROI compared to the overall estimation error. A small α indicates that the iterative solution is nearing convergence and sensor configuration should exploit the current ROI. Conversely, a large α indicates that the threat estimate with the current ROI is highly uncertain and the sensor configuration should provide exploration of the environment. When $\alpha = 0.5$ the SAMI reward function is the same as mutual information gain of the point i. We can optionally obtain the mutual information reward vector by calculating the reward on the set of vertices V as $\boldsymbol{\gamma} := \begin{bmatrix} \gamma(1) & \gamma(2) & \dots & \gamma(N_{\mathrm{g}}) \end{bmatrix}^{\mathsf{T}}$. We then formalize the information gain as a function of the sensor configuration as $\Gamma(\mathcal{C}_{\ell}) = \boldsymbol{\nu}^{\mathsf{T}} \boldsymbol{\gamma}$.

Next, we define a SAMI penalty function, which acts as FoV regularization, as the entropy of the measurement noise for a sensor configuration:

$$\Upsilon(\mathcal{C}_\ell) := -\frac{1}{2}\sum_{i\in\mathcal{F}}\left(\frac{1}{2}\ln(2\pi e) - \ln\sum_{k\in N_\mathrm{s}}(\boldsymbol{\nu}_k/\sigma_k^2)[\mathcal{F}]\right) \tag{6.7}$$

In Equation 6.7, we index the summation by $\mathcal{F}$ to ensure nonzero elements are removed prior to computing the elementwise entropy. Finally, the SAMI surrogate function is written as the difference between the reward and penalty terms:

$$S(\mathcal{C}_\ell) = \Gamma(\mathcal{C}_\ell) + \Upsilon(\mathcal{C}_\ell). \tag{6.8}$$

6.3 Sequential Optimization Optimality Guarantees

To find an optimal sensor configuration we maximize the SAMI surrogate objective function subject to spatial constraints, e.g., workspace limits on sensor placement and FoV radius bounds. Recall that the decision variables for optimization are $\{\boldsymbol{s}_1, \varrho_1, \boldsymbol{s}_2, \ldots, \varrho_{N_\mathrm{s}}\}$, i.e., a total of $3N_\mathrm{s}$ scalar variables in a 2D workspace. If the number of sensors N_s is moderate or high, this optimization problem is high-dimensional.

In previous chapters we have discussed different heuristic methods to mitigate the high dimensionality of the problem. In order to find the optimal sensor configuration $\mathcal{C}_\ell^*$ we first applied global optimization algorithms, such as evolutionary strategies and Bayesian optimization, for TDIG maximization. While these methods provided strong preliminary results in numerical studies performed, there was no guarantee of finding an optimal or near-optimal solution because the TDIG function does not have any convenient structural properties. These methods pursue *joint* optimization, i.e., finding all sensor configurations simultaneously.

Another method to mitigate the high dimensionality is *sequential* optimization, i.e., finding the optimal configuration for one sensor at a time. The main result of this chapter is that *the proposed SAMI surrogate function is submodular* (see Prop. 4). Therefore, the sequential optimization method is guaranteed to provide a near-optimal solution. Furthermore, we show that this method outperforms all the aforesaid TDIG maximization methods in terms of computational time and the self-adaptive property enables convergence in fewer iterations.

The TDIG metric is not submodular nor convex. However, we can say the following about the SAMI surrogate objective function.

Proposition 4. *The SAMI objective function S is a submodular function.*

Proof. The SAMI gain function Γ is a weighted coverage function of the union of vertices $\mathcal{F} := \cup_{k\in\{N_s\}}\mathcal{F}_k$ as each optimization of a k^{th} sensor either covers non-covered vertices or overlaps existing vertices at no additional reward. To help illustrate this, we decompose the sensor configuration $\mathcal{C}$ into the individual sensor FoVs $\mathcal{F}$ for the following discussion. Consider two sensors $i, j \in N_s$. For the case of disjoint field of views $\mathcal{F}_i \cap \mathcal{F}_j = \emptyset$ we obtain:

$$\begin{aligned}\Gamma(\mathcal{F}_i) + \Gamma(\mathcal{F}_j) &\geqslant \Gamma(\mathcal{F}_i \cup \mathcal{F}_j) + \Gamma(\mathcal{F}_i \cap \mathcal{F}_j)\\ \Gamma(\mathcal{F}_i) + \Gamma(\mathcal{F}_j) &= \Gamma(\mathcal{F}_i) + \Gamma(\mathcal{F}_j) + \emptyset\end{aligned}$$

For the case where the sensor field of views are intersecting such that $\mathcal{F}_i \cap \mathcal{F}_j \neq \emptyset$, let $\forall \mathcal{F} \subset V$, $\forall\, m, n \in V \backslash \mathcal{F}$ where $\Gamma(\{m\}) \geqslant \Gamma(\{n\})$ we can say:

$$\begin{aligned}\Gamma(\mathcal{F} \cup \{m\}) - \Gamma(\mathcal{F}) &\geqslant \Gamma(\mathcal{F} \cup \{m, n\}) - \Gamma(\mathcal{F} \cup \{m\})\\ \Gamma(\mathcal{F}) + \Gamma(\{m\}) - \Gamma(\mathcal{F}) &\geqslant \Gamma(\mathcal{F}) + \Gamma(\{m\}) + \Gamma(\{n\}) - \Gamma(\mathcal{F}) - \Gamma(\{m\})\\ \Gamma(\{m\}) &\geqslant \Gamma(\{n\})\end{aligned}$$

The equivalent definition of submodularity reduces to the initial inequality for any vertex not in the existing sensor FoV. The optimal sensor FoV found during sensor configuration will add $\{m\}$ to the existing set $\mathcal{F}$. As a result of the disjoint and intersecting sensor FoV cases, we may say the gain function of SAMI is submodular.

In assessing the penalty term, we first note diminishing returns associated with expanding FoV to cover more vertices. There first case that arises is that of disjoint sensor FoVs. In this case we say it satisfies sumbodularity as any $i, j \in N_\mathrm{s}$ exhibits:

$$\Upsilon(\mathcal{F}_i) + \Upsilon(\mathcal{F}_j) = \Upsilon(\mathcal{F}_i \cup \mathcal{F}_j)$$

The alternative case is when any sensor FoVs $i, j \in N_\mathrm{s}$ are intersecting. We first acknowledge that:

$$\Upsilon(\mathcal{F}_i) + \Upsilon(\mathcal{F}_j) = \Upsilon(\{\mathcal{F}_i \cup \mathcal{F}_j\} \backslash \{\mathcal{F}_i \cap \mathcal{F}_j\}) + 2\Upsilon(\mathcal{F}_i \cap \mathcal{F}_j)$$

Let $\Upsilon(\mathcal{F}_m) \leqslant \Upsilon(\mathcal{F}_i \cap \mathcal{F}_j)$ be the reduced cost function due to the weighted update from intersecting sensor covers in Equation 6.7. Due to the weighted sensor noise update for intersecting vertices, we can say that:

$$\Upsilon(\mathcal{F}_i \cup \mathcal{F}_j) + \Upsilon(\mathcal{F}_i \cap \mathcal{F}_j) = \Upsilon(\{\mathcal{F}_i \cup \mathcal{F}_j\} \backslash \{\mathcal{F}_i \cap \mathcal{F}_j\}) + \Upsilon(\mathcal{F}_i \cap \mathcal{F}_j) + \Upsilon(\mathcal{F}_m)$$

Given the previous statements, we can say that the SAMI penalty function exhibits the following condition for submodularity:

$$\Upsilon(\mathcal{F}_i) + \Upsilon(\mathcal{F}_j) \geqslant \Upsilon(\mathcal{F}_i \cup \mathcal{F}_j) + \Upsilon(\mathcal{F}_i \cap \mathcal{F}_j)$$

Finally, we show that the addition of these two submodular functions results in a submodular function as well:

$$\begin{aligned}
&(\Gamma + \Upsilon)(\mathcal{F}_i \cup \mathcal{F}_j) + (\Gamma + \Upsilon)(\mathcal{F}_i \cap \mathcal{F}_j) \\
=&(\Gamma(\mathcal{F}_i \cup \mathcal{F}_j) + \Gamma(\mathcal{F}_i \cap \mathcal{F}_j)) + (\Upsilon(\mathcal{F}_i \cup \mathcal{F}_j) + \Upsilon(\mathcal{F}_i \cap \mathcal{F}_j)) \\
\leqslant&(\Gamma(\mathcal{F}_i) + \Gamma(\mathcal{F}_j)) + (\Upsilon(\mathcal{F}_i) + \Upsilon(\mathcal{F}_j)) \\
=&(\Gamma + \Upsilon)(\mathcal{F}_i) + (\Gamma + \Upsilon)(\mathcal{F}_j)
\end{aligned}$$

Since S is an additive function of both the reward and penalty terms and each are submodular, we can say that S is submodular. □

Joint Optimization

Jointly optimizing all sensors and their parameters with the TDIG metric has a time complexity of $\mathcal{O}(N_{\mathrm{g}}^2)$ during each step in the sensor configuration optimization process. The SAMI metric has a time complexity of $\mathcal{O}(N_{\mathrm{g}})$ for both the information gain and penalty functions for each step in the optimization process. While the SAMI objective function requires taking the inverse both with $\mathcal{O}(\max\{\mathcal{R}, \mathcal{R}^c\}^3)$, this can be made a one time operation with batching for any vertex $i \notin \mathcal{R}$ (or $\mathcal{R}^c$) as in 8.1, or optimized in parallel otherwise with Equation 8.2. We note that during the sensor configuration optimization, the reward vector $\boldsymbol{\gamma}$ is only computed once prior to optimization rather than each step during the optimization. Optimizing the TDIG metric when the number of steps to achieve convergence is high can take more time than jointly optimizing the SAMI metric due to the reduced computational complexity of the iterative portion of optimization.

Of importance is the dimensionality of the joint optimization. The TDIG requires joint optimization of $N_d N_s$ parameters, where N_d is the dimensionality of each sensor. This can make achieving even a near globally optimal solution hard to achieve when the dimensionality is high. To counter these computational bottlenecks, we show next how the SAMI metric can be optimized sequentially with near-optimal guarantees.

Sequential Optimization

Corollary 3. *Sequential optimization of S is guaranteed to be at least $(1 - 1/e)$ the optimal value.*

The proof of Corollary 3 as a result of Proposition 4 is demonstrated in (Krause et al., 2008).

Given Corollary 3 we are able to say that sequentially optimizing the sensor configuration gives us a $(1 - 1/e)$ near-optimal solution. Sequential optimization reduces the dimensionality of the optimization process to N_d. Specifically, we reduce the high dimensional search space to a single sensor's configuration hyperparameters (position and FoV radius).

6.4 Performance Comparison

In this section, we provide computation results of implementing the CSCP iterative solution with both the TDIG metric as well as joint and sequential SAMI surrogate optimization. We numerically show the benefit of SAMI as a surrogate function for TDIG and that the sequential optimization of SAMI enables fast near-optimal sensor configuration with high performance.

In what follows, we performed a series of numerical experiments of 100 trials each with grid resolutions $N_{\mathrm{g}} \in \{11^2, 21^2, 31^2, 41^2, 51^2\}$ and $N_{\mathrm{s}} \in \{1, 3, 5, 7, 9\}$ sensors. The area of the environment was set to either 9, 25, or 49 square kilometers and the termination criteria was set to $\varepsilon = 0.1$. For experiments involving sampling path-plans, we took $N_a = 10$. The sensors were constrained to $\varrho^{min} = 0.05$ km and $\varrho^{max} = 0.5$ km.

6.4.1 TDIG vs. SAMI Performance

We begin by comparing the performance of the TDIG metric and sequential optimization of the SAMI surrogate function. Namely, we assess the iterations required for either method to achieve convergence below ε. We display the ratio of maximial sensor FoV area and environment area versus iterations to converge. We also track the average runtime for each method per iteration as a function of N_s.

In Figure 6.1, we compare the convergence of the TDIG and SAMI methods when we utilize just the current estimated optimal path-plan as our region of interest. The key takeaway is that the SAMI method has better performance for the majority of scenarios, especially when the maximum observability of the environment is low.

Next, we assessed the convergence of the TDIG and SAMI metrics with sampled path-plans. Figure 6.2 shows that SAMI method separates itself further in terms of increased performance, especially at low environment observability values.

Finally, Figure 6.3 shows the average runtime for both SAMI and TDIG sensor configuration maximization for path-plan and sampled path-plan regions of interest. This plot tells a couple stories. First, an obvious consequence of the sampled alternate path-plans is a nearly constant

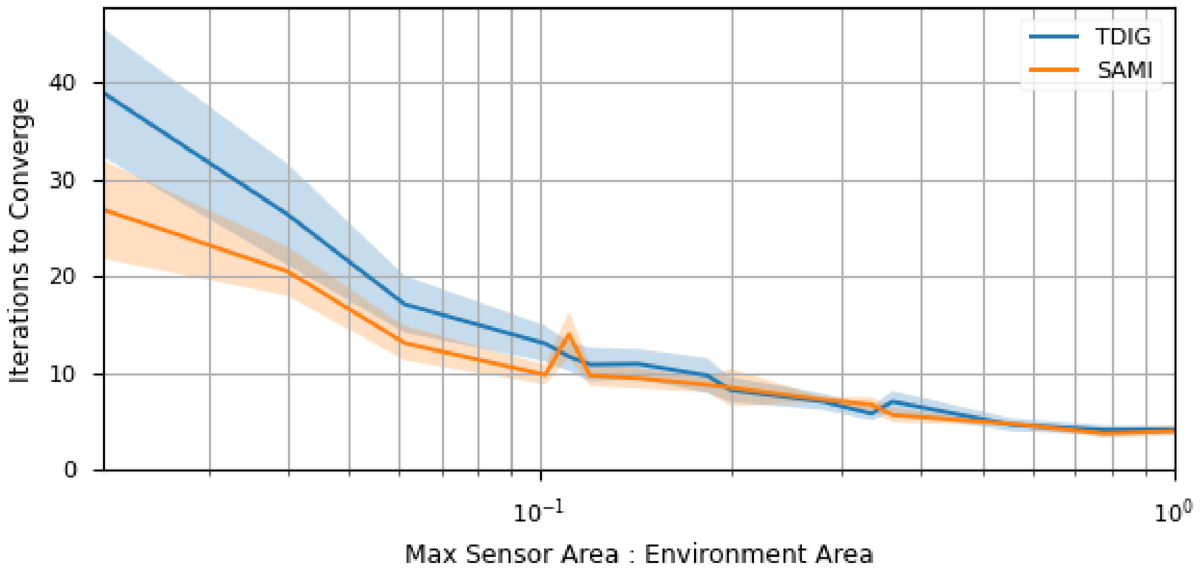


Figure 6.1: Comparison of SAMI vs. TDIG maximization for CSCP convergence utilizing just the current estimated optimal path plan. Assessed against the ratio between maximum observable sensor area and the area of the environment.

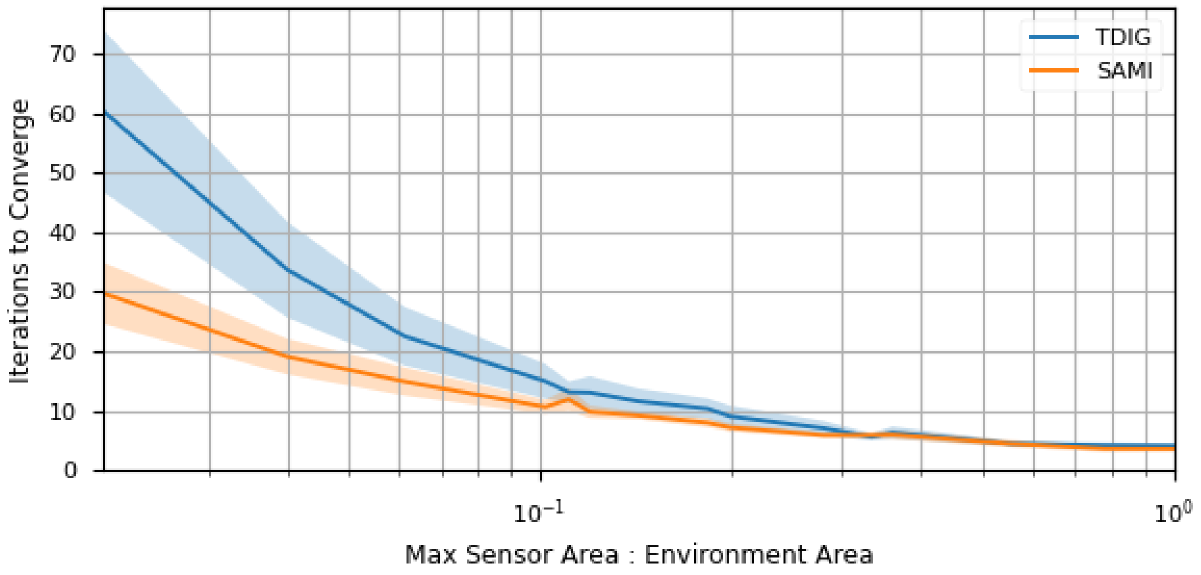


Figure 6.2: Comparison of SAMI vs. TDIG maximization for CSCP convergence utilizing statistically sampled path-plans. Assessed against the ratio between maximum observable sensor area and the area of the environment.

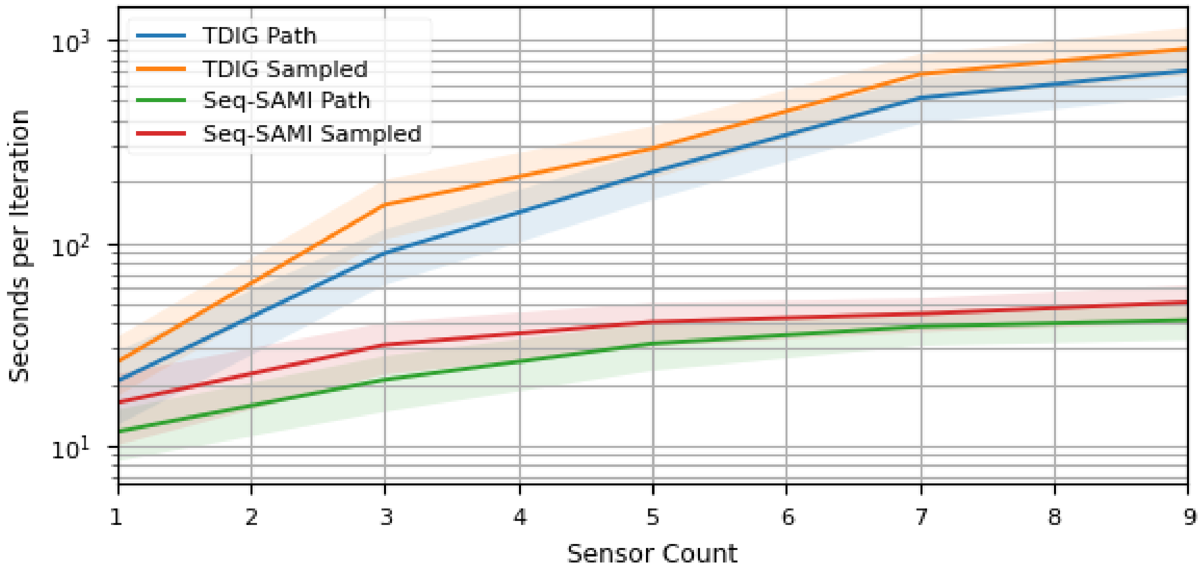


Figure 6.3: Comparison of runtime for both SAMI and TDIG maximization for both the current estimated optimal path-plan and sampled path-plans. Assessed against the ratio between maximum observable sensor area and the area of the environment.

offset in terms of increased computation time for both TDIG and SAMI. Second, the main result of this study is that the sequential optimization of the SAMI surrogate yields sensor configuration in orders of magnitude less time than TDIG, which is exemplified by this timing study. Using $N_s = 9$, SAMI sensor configuration iterations take nearly 1.5 orders of magnitude less time than the TDIG counterpart.

6.4.2 Joint vs. Sequential Sensor Configuration Optimization

Next, we compared sensor configuration optimization with both the joint and sequential SAMI strategies. Figure 6.4 depicts the iterations performance with a region of interest defined by the estimated optimal path-plan whereas Figure 6.5 shows the iteration performance when the region of interest is comprised of sampled path-plans. The main takeaway from both of these plots is that the joint optimization, while it should find a more optimal sensor configuration than sequential optimization, struggles to find the global optimal sensor configuration due to the high dimensionality of the problem. As a practical result, the sequential optimization of SAMI actually performs *better* and is more *reliable* at finding a near-optimal sensor configuration each iteration.

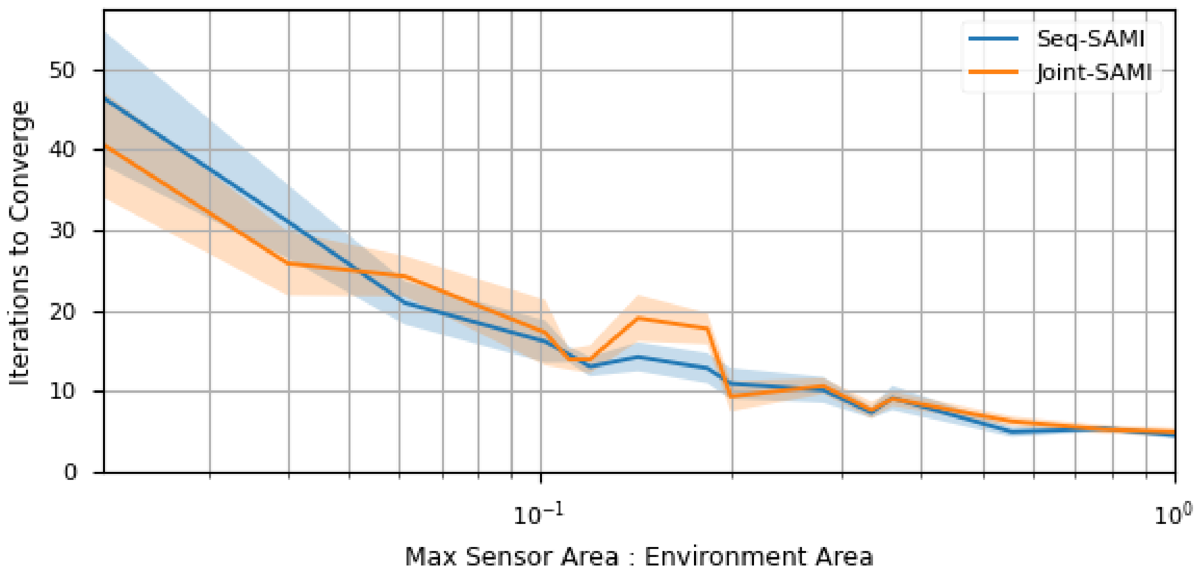


Figure 6.4: Comparison of convergence for Joint vs. Sequential SAMI maximization utilizing just the current estimated optimal path plan. Assessed against the ratio between maximum observable sensor area and the area of the environment.

We additionally assessed the runtime performance of the joint and sequential SAMI optimization as shown in Figure 6.6. To no surprise, the sequential SAMI optimization for both path-plan and sample path-plan region of interest cases vastly outperforms the joint optimization approach. In summary, practical real-world applications would benefit significantly by using the sequential SAMI sensor configuraiton otpimization given its iteration performance, it's reliability to find a near-optimal configuration, and it's fast runtime performance.

6.4.3 Effects of the Adaptation Parameter

Finally, we assessed the adaptation parameter α, which adjusts the balance of exploration and exploitation in the SAMI surrogate function. In our study, we compared the adaptive α to situations in which the adaptation parameter was fixed and unchanging. We used $\alpha \in \{1, 0.75, 0.5, 0.25, 0\}$ as the fixed parameter values. Notably, setting $\alpha = 1$ makes the algorithm value exploration explicitly outside the region of interest. Likewise, setting $\alpha = 0.5$ reduces the SAMI objective to maximizing information of the entire environment. Setting $\alpha = 0$ reduces the SAMI objective to a greedy strategy of sensor configuration which only cares about exploiting the region of interest.

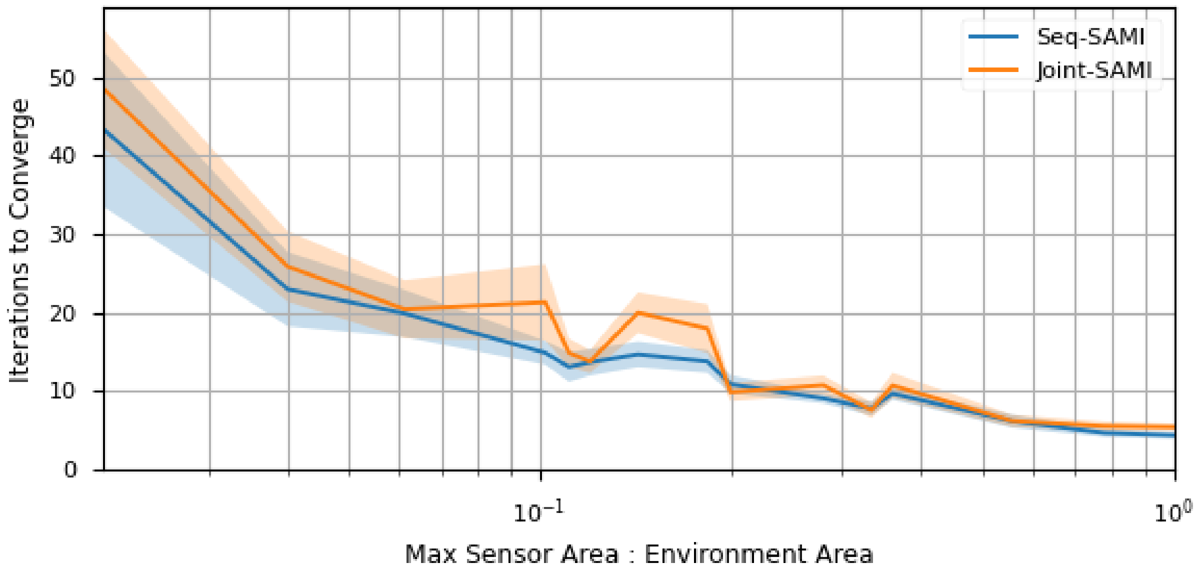


Figure 6.5: Comparison of convergence for Joint vs. Sequential SAMI maximization utilizing statistically sampled path-plans. Assessed against the ratio between maximum observable sensor area and the area of the environment.

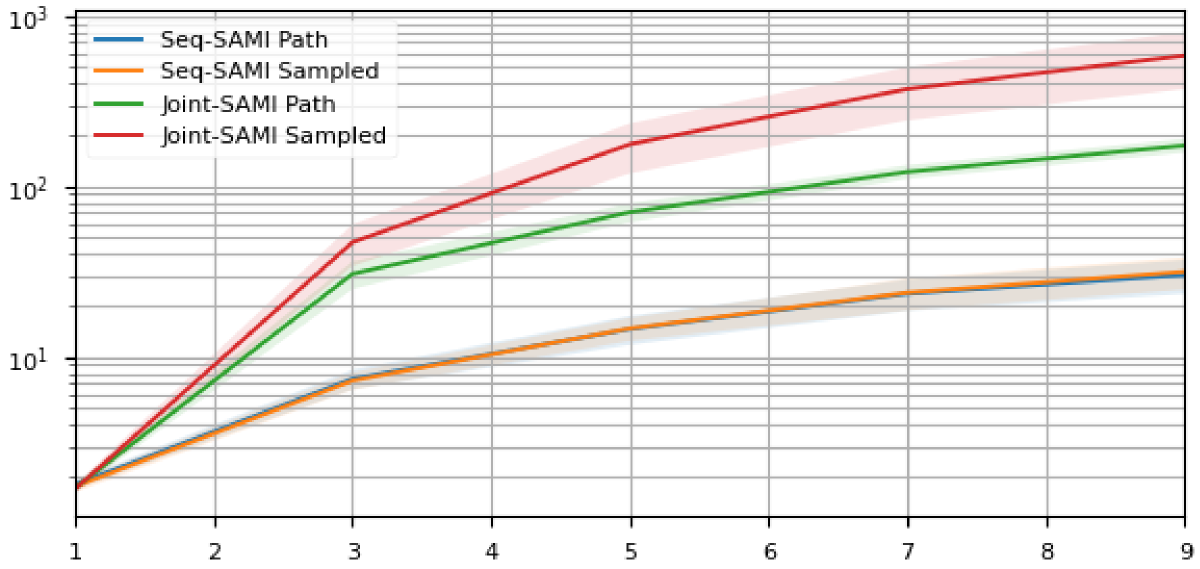


Figure 6.6: Comparison of runtime for Joint vs. Sequential SAMI maximization for both the current estimated optimal path-plan and sampled path-plans. Assessed against the ratio between maximum observable sensor area and the area of the environment.

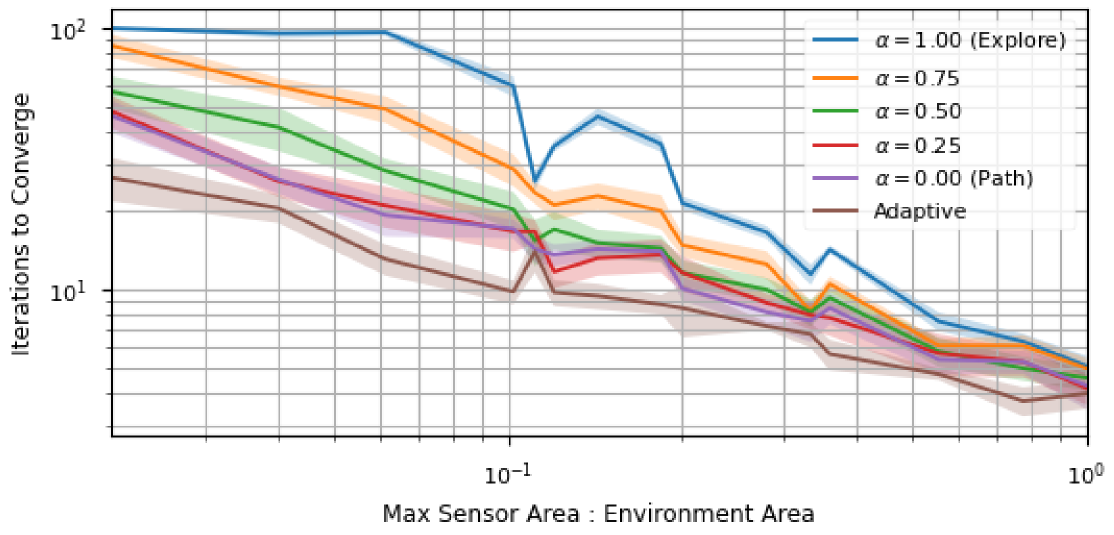


Figure 6.7: Average iterations to converge for various fixed α values and the self-adaptive α with sequential SAMI optimization.

The question we wish to answer from this study is: *is there a benefit to adapting α, or is a fixed policy, exploitative or exploratory or somewhere in between, more beneficial to performance?*

The answer to this question is easily gathered from the numerical results visualized in Figure 6.7. The beauty of this picture is the quite linear progression of improvement from pure exploration $\alpha = 1$ to pure exploitation $\alpha = 0$. In previous chapters, we showed that a task-driven approach (equivalent to $\alpha = 0$) outperformed greedy exploration of the environment $\alpha = 0.5$. These results reinforce this fact and further show that ignoring the region of interest is especially inferior (case of $\alpha = 1$). However, the most impressive result is that the adaptive α is able to outperform the entire range of fixed α parameters, demonstrating the self-adaptive component of SAMI, which balances exploration and exploitation given the current environment state, is pivotal to achieving ideal performance.

6.5 Demonstrative Example

To demonstrate the SAMI surrogate function for CSCP consider the example threat field and the optimal path-plan in Figure 6.8. For this example, the number of available sensors are $N_s = 5$ with maximum and minimum FoVs of $\varrho^{min} = 0.05$ and $\varrho^{max} = 0.5$. The problem domain

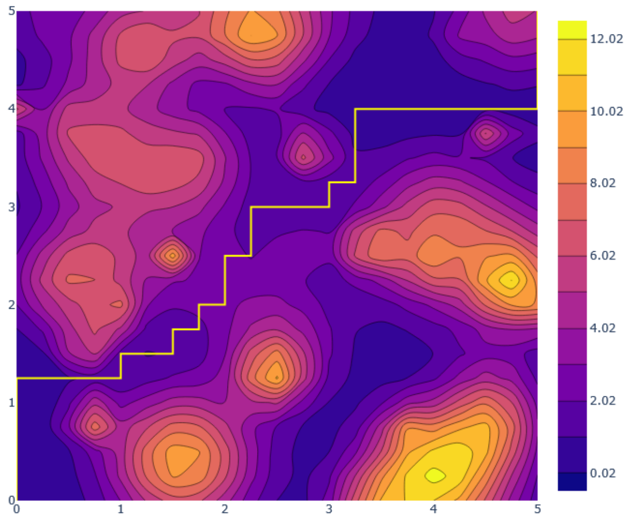
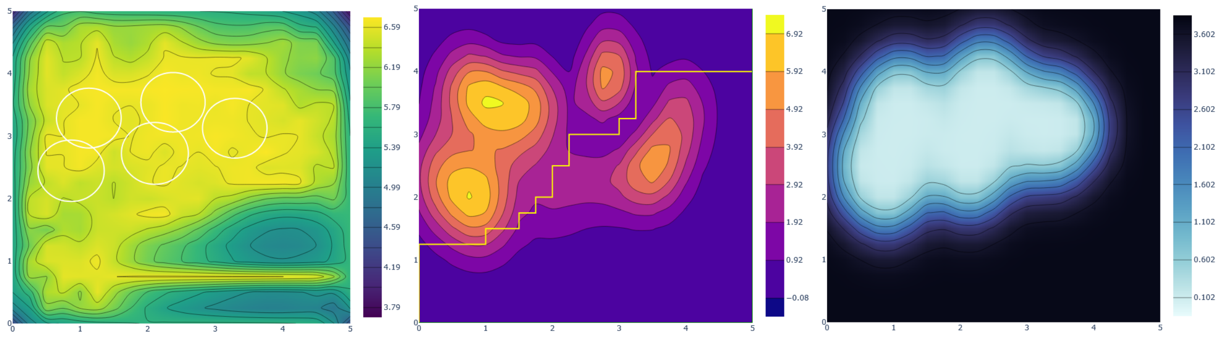


Figure 6.8: Threat field for the SAMI example along with the optimal path-plan (yellow).

workspace is $N_g = 21^2$ over the environment area of $25km^2$. In what follows, we illustrate the SAMI surrogate function applied to CSCP.

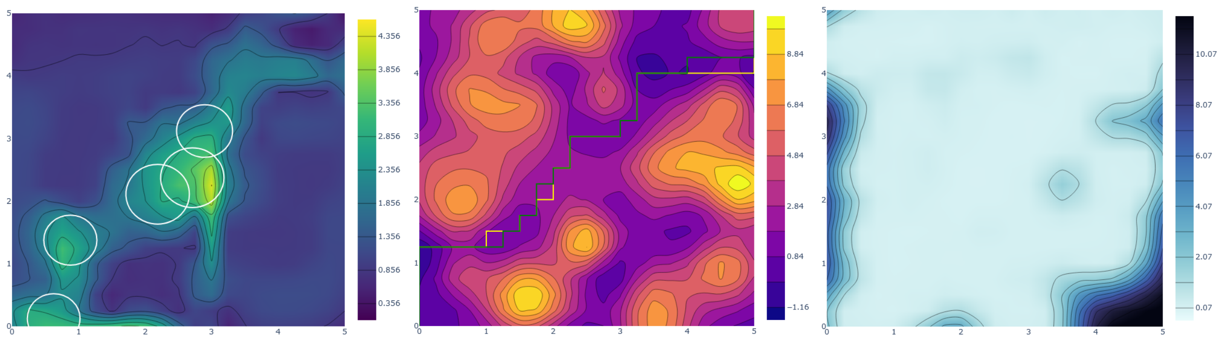
Figure 6.12 (a) depicts the fluxuation of the adaptation parameter over 10 iterations of CSCP with SAMI. Likewise, Figure 6.12 (c) shows the related average mutual information with respect to the environment and region of interest with each iteration. The adaptation parameter α decays and is proportional to the ratio of average mutual information with respect to the ROI and that of the environment. Figures 6.9, 6.10, 6.11 (a) depict the 2D representation of the SAMI reward term values for each vertex γ at iterations $\ell = 1, 5, 10$. It is visually apparent that as the adaptation parameter shrinks, the rewards converge to a tight area around the estimated path-plan. We note that in the 10^{th} iteration, the sensor FoVs become much tighter as the path-plan nears convergence due to the lower reward values and the regularization property of the SAMI cost term Υ.

Observing the estimated path-plan fit in Figures 6.9, 6.10, 6.11 (b), the CSCP method with SAMI is able to quickly find a near-optimal path-plan. This is further justified by inspecting Figure 6.12 (c) and (d) which show the current iteration path-plan path cost and variance. By the 7^{th} iteration the path-plan variance falls below $\varepsilon = 1$. Qualitatively, this example demonstrates the self-adaptive ability of SAMI to adapt from exploration-based sensor configuration to exploitation-based sensor configuration and while being able to optimize sensors configurations sequentially.



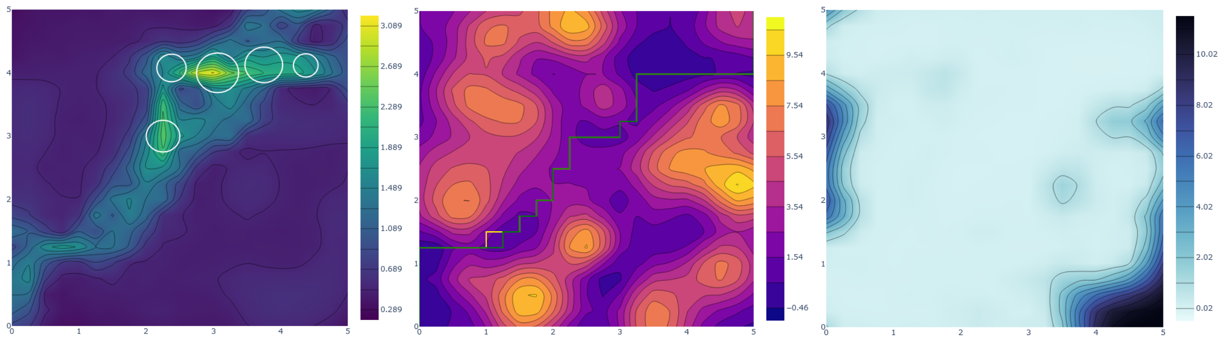
(a) (b) (c)

Figure 6.9: Sensor configuration FoVs (circles) overlaying the 2D SAMI reward values (a), the estimated threat field and estimated optimal path-plan (green) and true optimal path-plan (yellow) (b), and the threat error vertex variance values (c) at iteration $\ell = 1$.



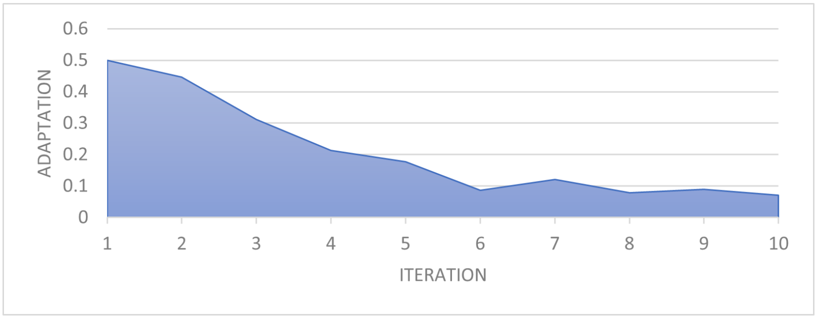
(a) (b) (c)

Figure 6.10: Sensor configuration FoVs (circles) overlaying the 2D SAMI reward values (a), the estimated threat field and estimated optimal path-plan (green) and true optimal path-plan (yellow) (b), and the threat error vertex variance values (c) at iteration $\ell = 5$.

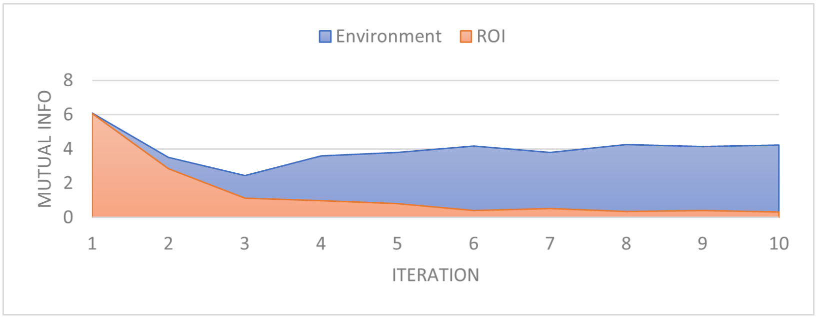


(a) (b) (c)

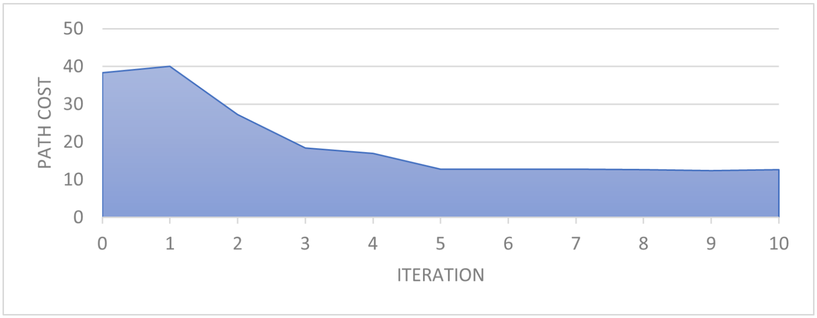
Figure 6.11: Sensor configuration FoVs (circles) overlaying the 2D SAMI reward values (a), the estimated threat field and estimated optimal path-plan (green) and true optimal path-plan (yellow) (b), and the threat error vertex variance values (c) at iteration $\ell = 10$.



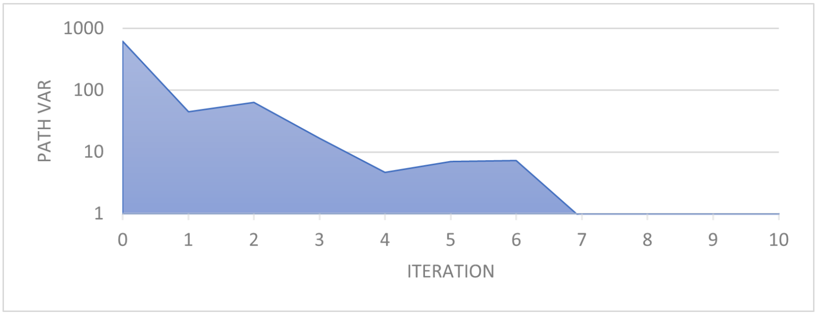
(a)



(b)



(c)



(d)

Figure 6.12: Adaptation coefficient α values (a), average mutual info for $\mathcal{R}$ and $\mathcal{R}^c$ (c), path cost (b), and path cost variance (d) for the demonstrative example over 10 iterations.

Chapter 7

CSCP in Multimodal Threat Environments

7.1 Problem Motivation

In many real world applications, the data that is collected from sensors are *heterogeneous*. In the previous chapters, we assumed that our sensor network of extroceptive sensors were homogeneous, meaning that the modality of the underlying threat environment was *unimodal*. In practice, heterogeneous sensor types can be utilized to capture various modalities of an environment which are only observable or partially observable by a particular sensor type. By utilizing heterogeneous sensor networks, we are able to characterize *multimodal* threat environments.

In previous chapters, we made an implicit assumption that each sensing agent had a payload with a *single* unimodal homogeneous sensor. However, there are many situations in which a sensing agent could have a payload with multiple heterogeneous sensor types, allowing for simultaneous sensing of variably correlated threat modalities in an environment. In what follows, we address situations in which a sensor network can be comprised of heterogeneous sensing agents with the following scenarios: (1) every sensing agent payload is equipped with every heterogeneous sensor, (2) the sensing agent payloads are deficient in at least one sensor type, and (3) the sensing agent payloads each contain a unique sensor type.

In this chapter, we address the heterogeneous sensor networks with these various sensing agent payload configuration scenarios. Furthermore, we build upon the sequential optimization results in Chapter 6 by developing an appropriate sequential sensor configuration strategy for dealing with heterogeneous payloads. Similarly, we update our statistical environment modeling method to accommodate the multimodal threat environment, enabling the cross-correlations of the threat modes to be determined.

7.2 Multimodal Threat Environment

In this section, we characterize the multimodal threat environment and it's difference from the unimodal threat environment detailed in previous chapters. First, we discuss the field of sensor fusion as it relates to the multimodal threat field. To help describe this topic, consider the following sensor types are available in our sensor network: (1) electro-optical (EO) imaging, (2) Infrared (IR) imaging, and (3) a Lidar (LI) point cloud. Fusion of these sensors could occur at the *sensor level* as raw data passes through circuitry, at the *data level* as the analog data acquisition becomes digitized, at the *feature level* where the data is combined through a latent embedded space, at the *decision level* which fuses independent output decisions from each sensor type, or at the *mission level* which fuses data with respect to spatial or task relevant correlations. In this work, we attempt to emulate mission level fusion via our statistical field estimation formulation. In terms of the aforementioned sensors, this would mean fusion occurs after the EO, IR, and LI sensors have been digitized, output a context-based decision about the data, and has been spatially modeled.

The *context* of the multimodal data fusion is of significant importance to how fusion is performed. We define each i^{th} threat field modality as $c^{(i)} : \mathcal{W} \to \mathbb{R}_{>0}$ as a strictly positive temporally static scalar field. We then define a *fused* threat field as $\check{c} := \boldsymbol{m}[1]c^{(1)} + \boldsymbol{m}[2]c^{(2)} + \cdots + \boldsymbol{m}[N_m]c^{(N_m)}$, where $\boldsymbol{m}$ is a user specified weighted fusion vector. The values prescribed to $\boldsymbol{m}$ define the context in which the fusion occurs. To be explicit, each i^{th} threat field modality $c^{(i)}$ represents the threat value associated with data obtainable from a particular sensor (i.e. thermal imaging or lidar) and the fused threat field represents the latent combined representation of the observable threats represented by each threat field mode $c^{(i)}$. A vector $\boldsymbol{m}$ is assigned by an expert in the field to assign relative importance of threat readings for a particular modality. In some cases, we may not have a linear combination of $\check{c}$, but some nonlinear mapping, which may be represented by a multimodal autoencoder which has a matrix of weights rather than the vector representation previously mentioned. For demonstration purposes for what follows, we will utilize the additive linear combination of threat modalities. The path cost is then calculated as $\mathcal{J}(\boldsymbol{\pi}) := \Delta p \sum_{j=1}^{\lambda} \check{c}(\boldsymbol{p}_{\boldsymbol{\pi}_j})$. Lastly, in what follows we make use of a multimodal vertex set $\tilde{V} := \{N_{\mathrm{g}} N_m\}$ to represent the vertex indices scaled to N_m modalities.

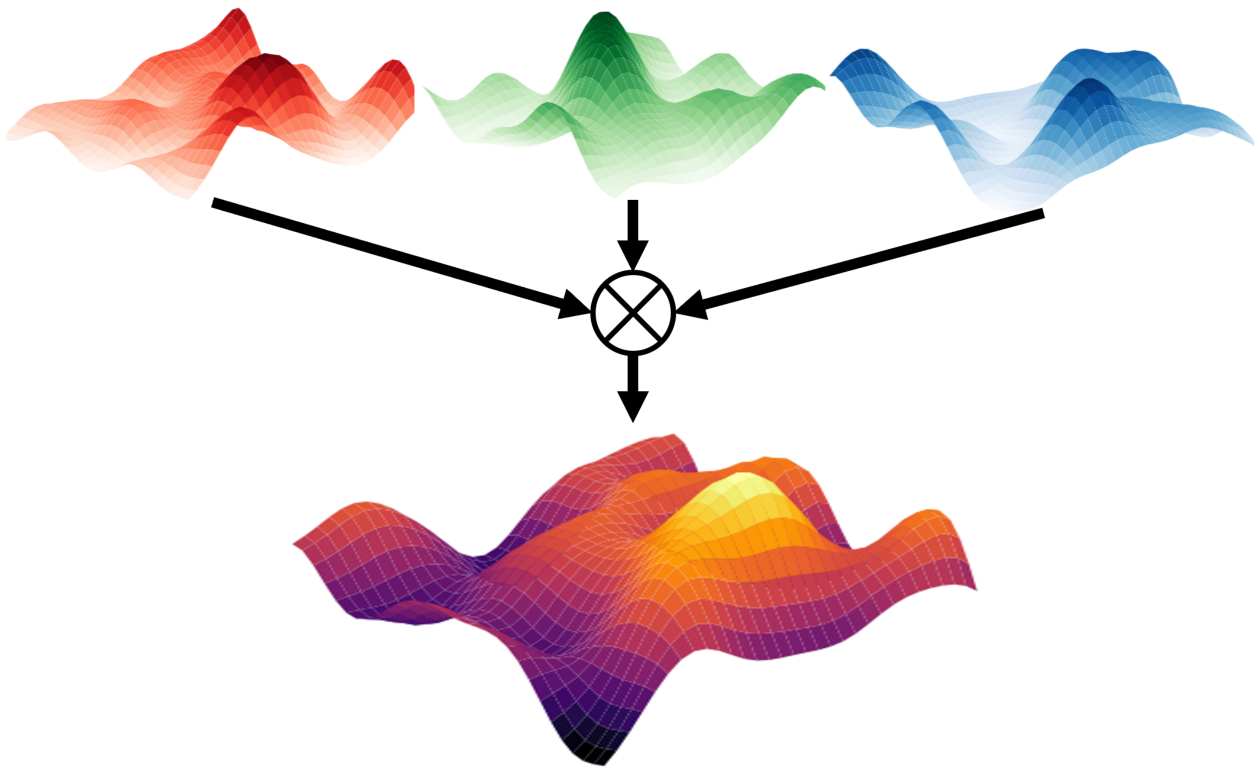


Figure 7.1: Pictorial Representation of Multimodal Threat Field Fusion

7.3 Heterogeneous Sensor Payloads

We begin this section by formalizing the nuances of heterogeneous sensor payloads within a sensor network. We denote the number of observable threat modalities by the sensor network as $N_m \in \mathbb{N}$. Next, we make the following assumption about each k^{th} sensing agent's field of view with respect to the N_m observable threat modalities.

Assumption 2. *We assume that the sensor network has a sensor payload with a field of view property that a multimodal field of view $\tilde{\mathcal{F}}_k \subset \tilde{V}$ has equivalent unimodal field of views such that $\mathcal{F}^{(1)} = \mathcal{F}^{(2)} = \cdots = \mathcal{F}^{(N_m)}$ for any $k \in N_s$.*

We introduce an observability incidence vector for each k^{th} sensor where $\boldsymbol{o}_k \in \{0,1\}^{N_m}$, which characterizes if the k^{th} sensor can view a threat modality and is specified by the user given each sensing agent's payload. We create an *observed* cover incidence matrix for each k^{th} sensor as $\tilde{\boldsymbol{\nu}}_k := \text{vec}(\boldsymbol{\nu}_k \boldsymbol{o}_k) \in \mathbb{R}^{N_g N_m \times 1}$, where we define $\text{vec}(\cdot) := \mathbb{R}^{N_g \times N_m} \to \mathbb{R}^{N_g N_m \times 1}$. Finally, the combined observed covered incidence vector becomes $\tilde{\boldsymbol{\nu}} := (\tilde{\boldsymbol{\nu}}_1 \vee \tilde{\boldsymbol{\nu}}_2 \vee \ldots \vee \tilde{\boldsymbol{\nu}}_{N_s})$.

We update the sensor observations notation as follows. The collection of sensor data locations for a particular i^{th} sensor type is denoted as $\boldsymbol{X}^{(i)} = \{\boldsymbol{x}_{11}^{(i)}, \ldots, \boldsymbol{x}_{km}^{(i)}, \ldots, \boldsymbol{x}_{N_{\mathrm{s}}M_{N_{\mathrm{s}}}}^{(i)}\} \; \forall \; i \in N_m$. We denote by, $\mathcal{X} = \{\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, \ldots, \boldsymbol{X}^{(N_m)}\}$, the set of sensor data locations for each sensing modality which is aggregated or updated with each iteration ℓ. The training matrix of the training set augmented by the corresponding modality index is formulated as follows:

$$\tilde{\boldsymbol{X}} := \begin{bmatrix} \boldsymbol{X}^{(1)} & \boldsymbol{X}^{(2)} & \ldots & \boldsymbol{X}^{(N_m)} \\ \mathbf{0} & \mathbf{1} & \ldots & \boldsymbol{N_m} \end{bmatrix}^{\intercal} \tag{7.1}$$

Similarly, we define the collection of vectorized noisy threat field observations and observation noise as $\tilde{\boldsymbol{z}} = \begin{bmatrix} \boldsymbol{z}_1^{\intercal} & \boldsymbol{z}_2^{\intercal} & \ldots & \boldsymbol{z}_{N_m}^{\intercal} \end{bmatrix}^{\intercal}$ and $\tilde{\boldsymbol{\sigma}} = \begin{bmatrix} \boldsymbol{\sigma_1}^{\intercal} & \boldsymbol{\sigma_2}^{\intercal} & \ldots & \boldsymbol{\sigma_{N_m}}^{\intercal} \end{bmatrix}^{\intercal}$, respectively. We say that each of the observations for each i^{th} modality is modeled as $\boldsymbol{z}_{km}^{(i)} = c^{(i)}(\boldsymbol{x}_{km}) + \eta_{km}^{(i)}$ and the measurement error η still follows Assumption 1 from Chapter 2, but for each i^{th} sensor modality.

7.4 Multimodal Field Estimation

The field estimation originally presented in Section 2.2.3 was for unimodal threat environments. We present modifications which enable multimodal field estimation, noting the continued use of Gaussian Process Regression. The joint distribution of Equation 2.8 is updated to account for the multimodal vectorized output and multimodal threat field estimates:

$$\begin{bmatrix} \tilde{\boldsymbol{z}} \\ \tilde{\boldsymbol{f}} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{K}_{\tilde{\boldsymbol{z}}} & \boldsymbol{K}_{*} \\ \boldsymbol{K}_{*}^{\intercal} & \boldsymbol{K}_{**} \end{bmatrix}\right) \tag{7.2}$$

We also update the kernel structure to enable learning of the cross-correlation between modalities. Recall the RBF-ARD kernel of Equation 2.6 is denoted as $\boldsymbol{K}^{R}$. To encode the auto-covariance and cross-covariance between the modalities, we utilize the intrinsic model of coregionalization (ICM) kernel $\boldsymbol{K}^{X}$ defined as:

$$\boldsymbol{K}^{X} = (\Theta_w \Theta_w^{\intercal} + \mathrm{diag}(\boldsymbol{\theta_v})) \tag{7.3}$$

The ICM kernel is a matrix of size $N_m \times N_m$ and has learnable parameters $\boldsymbol{\Theta_w}$ and $\boldsymbol{\theta_v}$. The matrix of parameters is of size $N_m \times r$, where $r \in \mathbb{N}$ is a small value to emulate low-rank positive definite correlation between modalities. The parameter vector $\boldsymbol{\theta}_v$ is of size $N_m \times 1$ and models the independent scaling factor of each modality. For performing statistical threat field modeling in this work, we utilize the kernel $\boldsymbol{K}_{ij} = \boldsymbol{K}^R_{ij} \cdot \boldsymbol{K}^X[i,j] \; \forall \; i,j \in N_m$. The resulting kernel has the form:

$$\boldsymbol{K} := K(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{X}}) = \begin{bmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{12} & \cdots & \boldsymbol{K}_{1N_m} \\ \boldsymbol{K}_{21} & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{K}_{N_m 1} & \cdots & \cdots & \boldsymbol{K}_{N_m N_m} \end{bmatrix}$$

The collection of hyperparameters to optimize are $\boldsymbol{\theta} = (\Theta_r, \Theta_w, \boldsymbol{\theta}_v)$ which can be optimized using Equation 2.7 and cross-validation. The multimodal input kernel with heteroscedastic noise vector is formulated as $\boldsymbol{K}_{\tilde{\boldsymbol{z}}} := \boldsymbol{K} + diag(\tilde{\boldsymbol{\sigma}})$. The diagonal elements of $\boldsymbol{K}$ represent the auto-covariance of points within a modality, whereas the off-diagonal elements represent the cross-covariance which models the latent relationship between modalities.

From the joint distribution, we can obtain the current iteration multimodal threat field estimate and multimodal threat error covariance matrix as:

$$\tilde{\boldsymbol{f}}_\ell = \boldsymbol{K}_*^\intercal \boldsymbol{K}_{\tilde{\boldsymbol{z}}}^{-1} \tilde{\boldsymbol{z}} \tag{7.4}$$

$$\tilde{\boldsymbol{P}}_\ell = \boldsymbol{K}_{**} - \boldsymbol{K}_*^\intercal \boldsymbol{K}_{\tilde{\boldsymbol{z}}}^{-1} \boldsymbol{K}_* \tag{7.5}$$

We note that the multimodal threat field estimate and multimodal threat field error covariance matrix are constructed as:

$$\tilde{\boldsymbol{f}}_\ell = \begin{bmatrix} \boldsymbol{f}_\ell^{(1)\intercal} & \boldsymbol{f}_\ell^{(2)\intercal} & \dots & \boldsymbol{f}_\ell^{(N_m)\intercal} \end{bmatrix}^\intercal$$

$$\tilde{\boldsymbol{P}}_\ell = \begin{bmatrix} \boldsymbol{P}_\ell^{(11)} & \boldsymbol{P}_\ell^{(12)} & \cdots & \boldsymbol{P}_\ell^{(N_m)} \\ \boldsymbol{P}_\ell^{(21)} & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{P}_\ell^{(N_m 1)} & \cdots & \cdots & \boldsymbol{P}_\ell^{(N_m N_m)} \end{bmatrix}$$

Given the predefined context of our fusion of threat modalities, we utilize the weighted fusion vector $\boldsymbol{m} \in \mathbb{R}^{N_m}$ and fuse the multimodal predictions. The fused threat field estimate and fused threat field error covariance matrix can then be computed as:

$$\check{\boldsymbol{f}}_\ell = \sum_{i=1}^{N_m} \boldsymbol{m}[i]\tilde{\boldsymbol{f}}_\ell^{(i)} \tag{7.6}$$

$$\check{\boldsymbol{P}}_\ell = \sum_{i=1}^{N_m} \boldsymbol{m}^2[i]\tilde{\boldsymbol{P}}_\ell^{(ii)} + 2\sum_{j=2}^{N_m}\sum_{k=1}^{j-1} \boldsymbol{m}[j]\boldsymbol{m}[k]\tilde{\boldsymbol{P}}_\ell^{(jk)} \tag{7.7}$$

We may then use the fused threat field and fused threat error covariance matrix to find the estimated optimal path-plan and the estimated path-plan variance.

7.5 Sensor Configuration for Heterogeneous Sensor Networks

The introduction of heterogeneous sensors presents some hidden nuances in addition to the aforementioned modifications to the CSCP algorithm. Namely, the sensor network is capable of handling heterogeneous payloads on each sensing agent. This presents three unique situations in which the sensor configuration problem can be formalized. In what follows, we describe situations in which every sensing agent is equipped with every sensor type, a mixture of sensor types, or only a single sensor type. We show how to apply the SAMI-based surrogate optimization function from Chapter 6 to enable near-optimal sequential optimization for each case.

We note a few notation updates. First, we say that a multimodal region of interest is defined as the collection of region of interest vertices with index values biased by the appropriate mode index as $\tilde{\mathcal{R}} := \{\cup_{i=0}^{N_m-1}\mathcal{R} + iN_{\mathrm{g}}\}$. The multimodal region of interest is solely used for indexing in the SAMI formulation. We can find the entropy of any vertex $i \in \tilde{V}$ as:

$$h(i) := \frac{1}{2}\ln(2\pi e\tilde{\boldsymbol{P}}_\ell[i,i]) \tag{7.8}$$

Recall from Chapter 6 that in certain cases the conditional entropy computations can be batched and computed in parallel. For any $i \notin \mathcal{R}$, the batched conditional entropy vector $\boldsymbol{h}(\cdot|\tilde{\mathcal{R}}_{\setminus i})$

is computed as:

$$\boldsymbol{h}(\cdot|\tilde{\mathcal{R}}_{\setminus i}) = \frac{1}{2}\ln((2\pi e)diag(\tilde{\boldsymbol{P}}_\ell - \tilde{\boldsymbol{P}}_\ell[\cdot,\tilde{\mathcal{R}}]\tilde{\boldsymbol{P}}_\ell[\tilde{\mathcal{R}},\tilde{\mathcal{R}}]^{-1}\tilde{\boldsymbol{P}}_\ell[\tilde{\mathcal{R}},\cdot])) \tag{7.9}$$

We note that the matrix inverse only needs to be computed once. However, for any $i \in \mathcal{R}$, the conditional entropy of any vertex $i \in \{N_g N_m\}$ given $\tilde{\mathcal{R}}_{\setminus i}$ can be computed in parallel as:

$$h(i|\tilde{\mathcal{R}}_{\setminus i}) = \frac{1}{2}\ln((2\pi e)(\tilde{\boldsymbol{P}}_\ell[i,i] - \tilde{\boldsymbol{P}}_\ell[\tilde{\mathcal{R}}_{\setminus i},i]^\intercal\tilde{\boldsymbol{P}}_\ell[\tilde{\mathcal{R}}_{\setminus i},\tilde{\mathcal{R}}_{\setminus i}]^{-1}\tilde{\boldsymbol{P}}_\ell[\tilde{\mathcal{R}}_{\setminus i},i])) \tag{7.10}$$

Given these equations, we can calculate the mutual information I between the multimodal region of interest and any vertex for a particular mode $i \in \tilde{V}$ as:

$$I(\tilde{\mathcal{R}}_{\setminus i}; i) := h(i) - h(i|\tilde{\mathcal{R}}_{\setminus i}). \tag{7.11}$$

The reward term for multimodal threat fields is then adapted to be:

$$R(i) := (1-\alpha)I(\tilde{\mathcal{R}}_{\setminus i}; i) + \alpha I(\tilde{\mathcal{R}}^c_{\setminus i}; i) \tag{7.12}$$

In Equation 7.12, the ROI complement is taken as $\tilde{\mathcal{R}}^c := \tilde{V}\setminus\tilde{\mathcal{R}}$ and $i \in \tilde{V}$. We also update the mutual information reward vector to be $\boldsymbol{\gamma} := \begin{bmatrix}\gamma(1) & \gamma(2) & \ldots & \gamma(N_\text{g}) & \ldots & \gamma(N_\text{g}N_m)\end{bmatrix}^\intercal$. The reward function given the sensor configuration is $\Gamma(\mathcal{C}_\ell) = \tilde{\boldsymbol{\nu}}^\intercal\boldsymbol{\gamma}$.

The SAMI penalty function is calculated as:

$$\Upsilon(\mathcal{C}_\ell) := -\frac{1}{2}\sum_{i\in\mathcal{F}}\left(\frac{1}{2}\ln(2\pi e) - \ln\sum_{k\in N_\text{s}}(\boldsymbol{\nu}_k \odot \boldsymbol{\sigma}_k^{-2})[\tilde{\mathcal{F}}]\right) \tag{7.13}$$

In Equation 7.13, the $\odot$ operator is the Hadamard product, which is used to perform element-wise multiplication between the cover incidence vector and the inverse noise vector $\boldsymbol{\sigma_k}^{-2}$ which is defined as $\boldsymbol{\sigma_k}^{-2}[i] = 0$ if $\boldsymbol{o}[i] = 0$ and $\boldsymbol{\sigma_k}^{-2}[i] \propto \varrho_k$ otherwise, for $i \in N_m$. The entries are indexed by $[\tilde{\mathcal{F}}]$ to ensure only covered vertices are accounted for prior to taking the natural logarithm of each element. Finally, the SAMI surrogate optimization function is the same as Equation 6.8 and by consequence the sequential optimization properties. Next, we describe the nuances of optimizing the SAMI objective function for the various sensor payload configuration cases.

7.5.1 Fully Observable Sensor Payloads

The first case we address is when all sensing agents within the sensor network are equipped with all sensor types of interest to observe the threat field, i.e. $\boldsymbol{o}_k = \mathbf{1} \; \forall \; k \in N_\mathrm{s}$. Due to Assumption 2, this problem reduces to maximizing the SAMI surrogate objective function directly over multimodal SAMI surrogate values. As such the optimization problem can sequentially optimize sensor configuration for each set of k^{th} sensor parameters $\{\boldsymbol{s}_k, \varrho_k\}$ to find $\mathcal{C}^*_\ell := \arg\max S(\mathcal{C}_\ell)$.

7.5.2 Partially Observable Sensor Payloads

The next possible sensor configuration scenario is where the sensor network has agents with mixed observability of the multimodal threat field. As such, the observability vector for the sensors is such that $1 \leqslant \sum_{i \in N_m} \boldsymbol{o}_k \leqslant N_m$ for all $k \in N_s$. As a result of this mixed partially observable sensor payload scenario, a question naturally arises: *how do we ensure that the sensor configuration remains submodular?*

To verbosely illustrate the reason why this question is important consider our sensor's from earlier in the chapter. Consider that we have available three sensing agent's $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3\}$ with the following payloads: $\mathcal{S}_1 = \{\text{EO, IR}\}$, $\mathcal{S}_2 = \{\text{EO, LI}\}$, and $\mathcal{S}_3 = \{\text{IR, LI}\}$. To determine the sequence in which a sensing agent in the sensor network is optimized to adhere to the submodularity property, we need to calculate the total reward for each modality as $\bar{\gamma}_i = \sum_{j \in N_\mathrm{g}} (1 - \tilde{\boldsymbol{\nu}}^{(i)}) \boldsymbol{\gamma}^{(i)}$ prior to each sequential sensor configuration optimization. We denote the $\tilde{\boldsymbol{\nu}}^{(i)}$ as the partition of the multimodal sensor cover incidence for the i^{th} modality. Similarly, the i^{th} modality partition of the SAMI reward vector is denoted $\boldsymbol{\gamma}^{(i)}$. The total rewards is then $\bar{\boldsymbol{\gamma}} = \begin{bmatrix} \bar{\gamma}_1 & \bar{\gamma}_2 & \dots & \bar{\gamma}_{N_m} \end{bmatrix}^\mathsf{T}$. We may then calculate the total potential reward for each k^{th} sensor as $\bar{\gamma}^{(k)} = \boldsymbol{o}_k \bar{\boldsymbol{\gamma}}$. Therefore, each iteration of sequential sensor configuration, we choose $\mathcal{S}_k = \arg\max \bar{\gamma}^{(k)}$, optimize it with the SAMI objective function, and remove it from the set $\mathcal{S}$ (just for that round of optimization). In our example, if $\bar{R}^{(2)}$ was the maximum value, we would perform sensor configuration optimization with $\mathcal{S}_2$ and remove it from the set of optimizable sensing agents, leaving only $\mathcal{S}_1$ and $\mathcal{S}_3$ to be optimized. We proceed to score the sensing agents again until they have all been optimized.

7.5.3 Uniquely Observable Sensor Payloads

Finally, we consider the situation in which each sensing agent in the sensor network $\mathcal{S}$ has only one modality of observability, i.e. $\sum_{i \in N_m} \boldsymbol{o}_k = 1$ for all $k \in N_m$. This is a special observability case which has unique sensor configuration optimization implications. In this scenario, consider each sensing agent within the sensor network can only observe one modality. We no longer require ranking the sensor modalities as they are entirely separable and additive. Due to this, the sensor configuration for uniquely observable sensor payloads allows for parallelization of sequential sensor configuration for each modality. In fact, each mode reduces to solving independent sensor configuration optimization N_m times in parallel, allowing for computational savings.

7.6 Results and Discussion

Many unmanned aerial vehicles, commercial and industrial, have various sensor payload configuration options. 'Fancy' UAVs can come equipped with multiple sensor peripherals at a premium price. To handle multiple sensors the UAV needs to be engineered to handle the additional weight and be structurally capable of hosting them. In contrast, there are many specialized UAVs which come equipped with single sensor peripherals such as EO imaging, infrared, or similar. In what follows, we attempt to address the justification of expensive UAV payloads which carry multiple sensor payloads rather than multiple unimodal UAVs as it relates to path-planning.

We conducted a study with four various sensor networks in an environment of area $9km^2$, workspace resolution of 21^2, and a desired termination threshold $\varepsilon = 1$. We considered a multimodal threat field with $N_m = 3$ correlated threat modalities. All mobile sensing agents were constrained to $\varrho^{\min} = 0.05$ and $\varrho^{\max} = 0.5$. The region of interest for the experiments was found with $N_a = 3$ alternate path-plans.

We considered four various sensor network scenarios for the experiments. The first sensor network $\mathcal{S}_A$ was comprised of 3 fully observable sensors. $\mathcal{S}_B$ had three sensors with pairs of observability such that each modality was observable by at least 2. It also included a single fully observable sensor totaling $N_s = 4$. Sensor network $\mathcal{S}_C$ had the same pairs of observability as $\mathcal{S}_B$, but instead of the single fully observable sensing agent, it had three unimodal sensors (one for each modality) totaling $N_s = 6$. Finally, we considered a sensor network $\mathcal{S}_D$ which was comprised of $N_s = 9$ unimodal sensors such that each modality equally had 3 sensors which could make observations. These networks were chosen to study the effect of modifying the degrees of freedom of the sensor network on convergence and sensor configuration optimization search time. The average iterations for convergence and the average sensor configuration optimization time for each sensor network are recorded in Table 7.1. We make the following observations from the collected results. In summary, the parallel optimization of sensor networks with unique observability, by way of the separable nature of the multimodal formulation of the SAMI surrogate function, are able to quickly find sensor configurations whilst providing good convergence performance.

Sensor Network	Iterations	Configuration Time
Network $\mathcal{S}_A$	16.331 ± 4.163	2.313 ± 0.683
Network $\mathcal{S}_B$	16.033 ± 3.436	3.068 ± 0.995
Network $\mathcal{S}_C$	15.667 ± 4.619	4.900 ± 1.472
Network $\mathcal{S}_D$	14.330 ± 4.041	2.878 ± 1.004

Table 7.1: Average results of the comparative study for various sensor payload configurations

Increasing sensor network observability flexibility improves convergence performance

The observation flexibility refers to the constraint of having identical FoVs to optimize on a single sensor payload. The observation flexibility's for the experiment sensor networks are in increasing order $\mathcal{S}_A < \mathcal{S}_B < \mathcal{S}_C < \mathcal{S}_D$. The results show that adding the increased flexibility by not confining multiple multimodal sensor payloads to a single sensing agent proportionally decreases the average iterations for path-plan convergence. This answers our initial quandry between buying an expensive UAV equipped with many heterogeneous sensors versus buying multiple inexpensive UAVs. For coupled sensing and path-planning problems the additional flexibility by distributing the sensors across sensing agents within the network is more valuable.

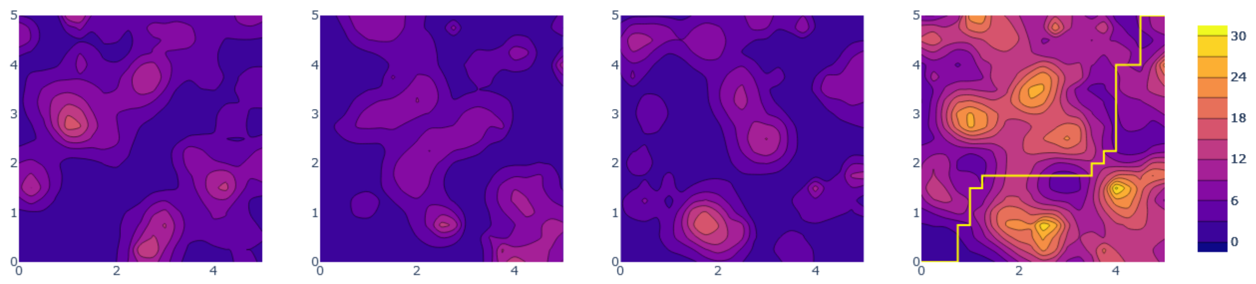


Figure 7.2: Ground truth threat field modalities and the fused threat field (right) along with the optimal path-plan.

Parallel optimization of unimodal sensor payloads yields low configuration times

Somewhat unsurprisingly, the parallel sequential optimization of sensor network $\mathcal{S}_D$ which had sensing agents with unimodal observability, was able to be optimized faster than sensor networks $\mathcal{S}_B$ and $\mathcal{S}_C$. We note that the reason it still takes longer to configure these sensors over sensor network $\mathcal{S}_A$ is that there are more potential configurations and globally optimizing is made more difficult. However, we note how close the runtime performance is for sensor configuration between $\mathcal{S}_A$ and $\mathcal{S}_D$ and note that the time difference is negligible in comparison to the average iterations until convergence performance gain.

7.7 Demonstrative Example

In what follows, we demonstrate the MM-CSCP algorithm on an example randomly generated multimodal threat field with $N_m = 3$. The problem definition remains the same as the numerical experiments from the previous section, but we increase the environment area to $25km^2$. The individual threat field modalities and the fused threat field along with the optimal path-plan are shown in Figure 7.2. We considered a sensor network with 3 sensing agents with unique pairings of observability and 2 sensing agents with unimodal observability for the first two threat modes. Therefore, the first two threat modes have up to three FoV covers per iteration while the third only has two. The initial sensor network configuration is depicted overlaying the SAMI reward function for each vertex in Figure 7.3.

The data collected from the first iteration sensor configuration is then utilized to determine

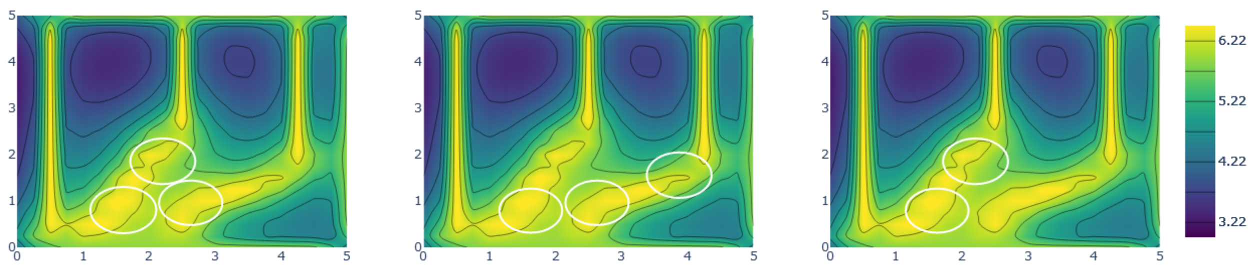


Figure 7.3: Initial iteration sensor network configurations overlaying the multimodal SAMI reward values for each threat mode.

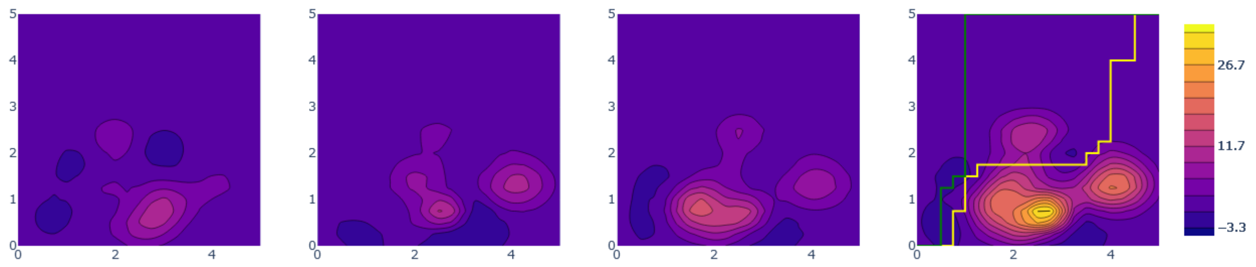


Figure 7.4: Initial multimodal threat field estimates and fused threat field estimate (right) along with the estimated optimal path-plan (green) and the true optimal path-plan (yellow).

the multimodal threat field estimate and subsequently the fused threat field estimate as shown in Figure 7.4. The resultant estimated path-plan is shown in this figure, and Figure 7.5 shows the corresponding multimodal threat error covariance values for each vertex and the fused covariance values. The algorithm proceeds until convergence at iteration $\ell = 22$. Figures 7.6 through 7.17 show the SAMI reward functions and sensor configurations, the multimodal and fused threat field estimates with the estimated path-plan, and the multimodal and fused threat variances at select iterations.

The SAMI reward examples show that in some cases sensing agents with paired observability are configured such that the reward of one modality is considered in conjunction with a potentially low reward modality. We also note that the multimodal threat field estimation and fusion visually does a good job at learning the environment. We also observe from the final fused threat error variance in Figure 7.17 that the MM-CSCP still emphasizes learning task-driven regions of interest, rather than the entire environment, in order to make the path-plan decision. Ultimately, the estimated path-plan finds a near-optimal path-plan within a multimodal threat environment.

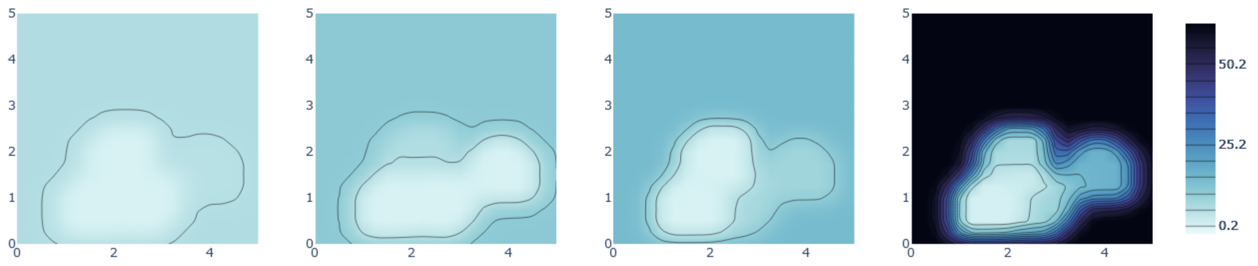


Figure 7.5: Initial multimodal threat field error covariance vertex values and fused vertex covariance values.

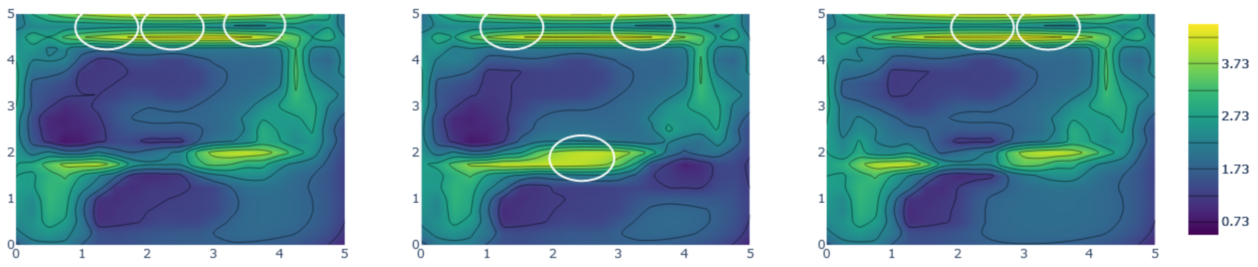


Figure 7.6: Sensor network configurations overlaying the multimodal SAMI reward values for each threat mode at $\ell = 3$.

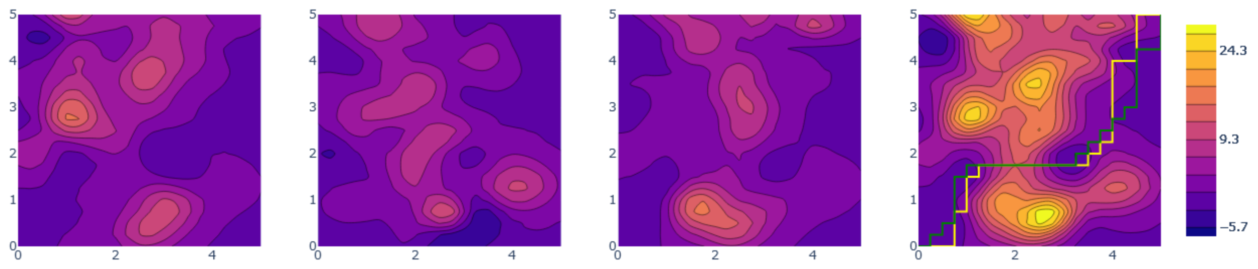


Figure 7.7: Multimodal threat field estimates and fused threat field estimate (right) along with the estimated optimal path-plan (green) and the true optimal path-plan (yellow) at iteration $\ell = 3$.

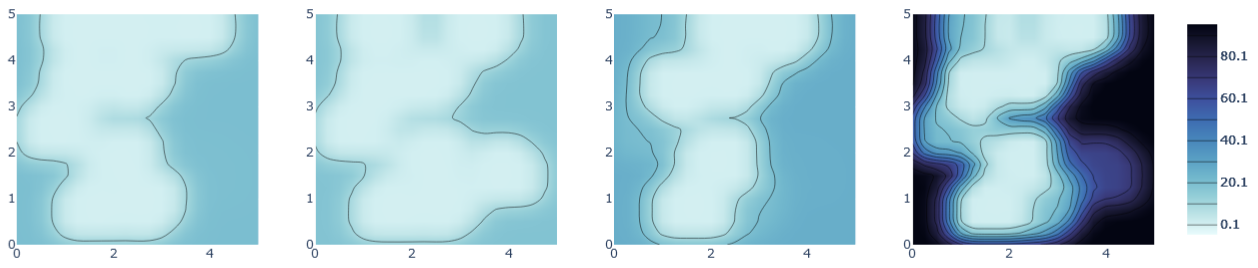


Figure 7.8: Multimodal threat field error covariance vertex values and fused vertex covariance values at iteration $\ell = 3$.

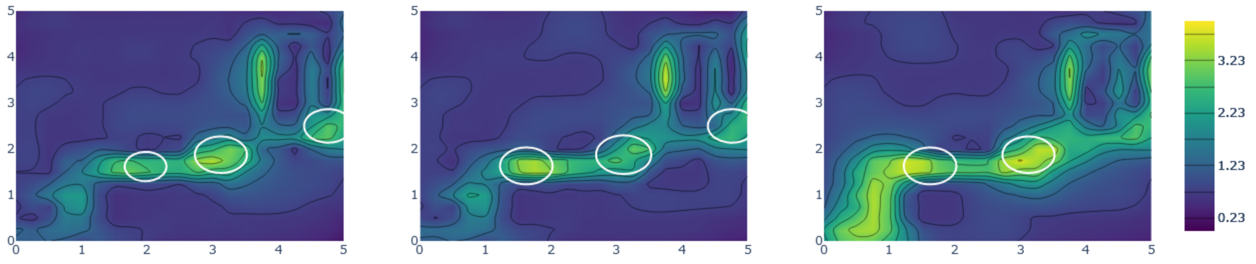


Figure 7.9: Sensor network configurations overlaying the multimodal SAMI reward values for each threat mode at $\ell = 7$.

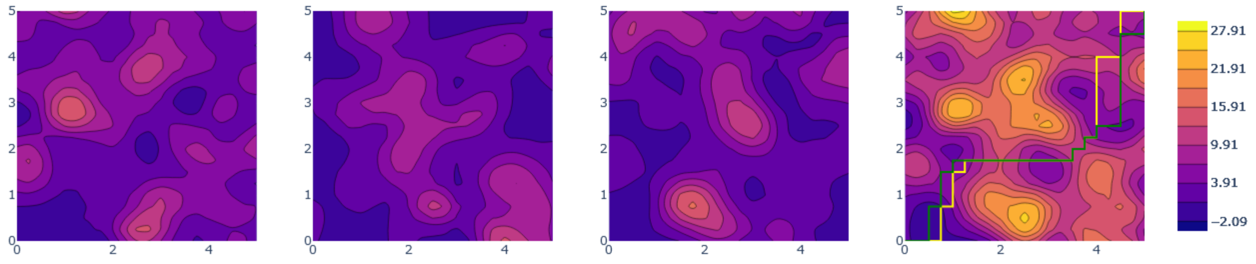


Figure 7.10: Multimodal threat field estimates and fused threat field estimate (right) along with the estimated optimal path-plan (green) and the true optimal path-plan (yellow) at iteration $\ell = 7$.

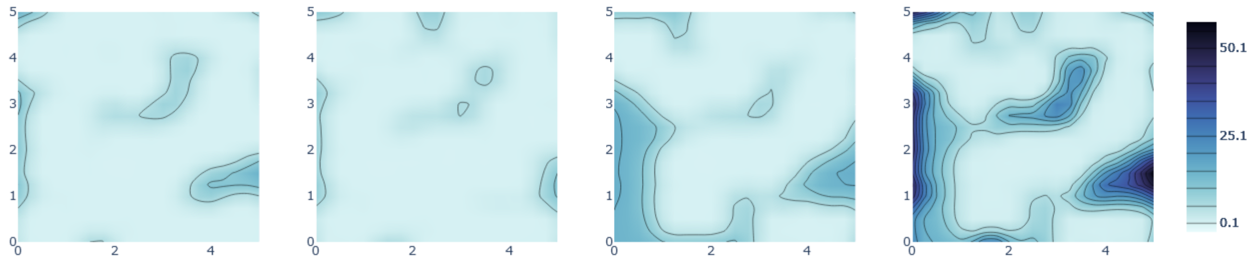


Figure 7.11: Multimodal threat field error covariance vertex values and fused vertex covariance values at iteration $\ell = 7$.

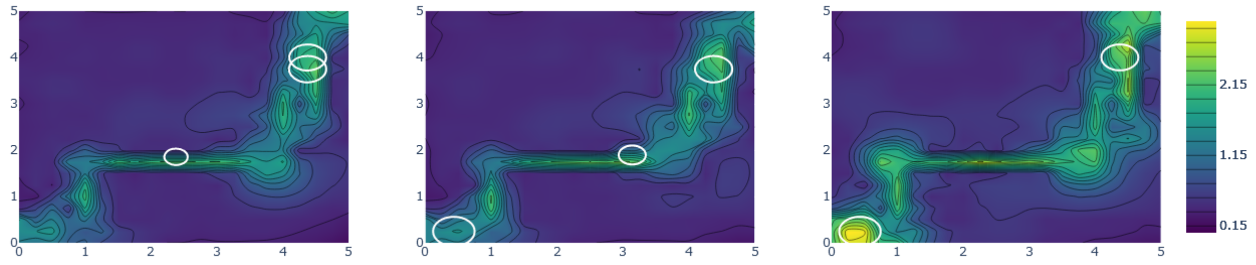


Figure 7.12: Sensor network configurations overlaying the multimodal SAMI reward values for each threat mode at $\ell = 13$.

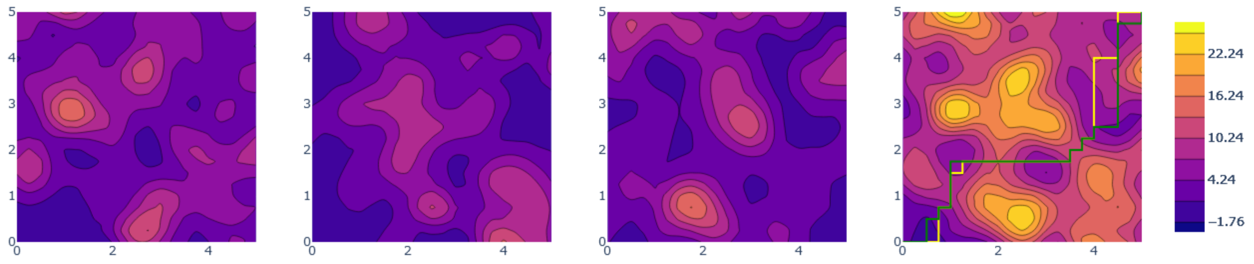


Figure 7.13: *Multimodal threat field estimates and fused threat field estimate (right) along with the estimated optimal path-plan (green) and the true optimal path-plan (yellow) at iteration $\ell = 13$.*

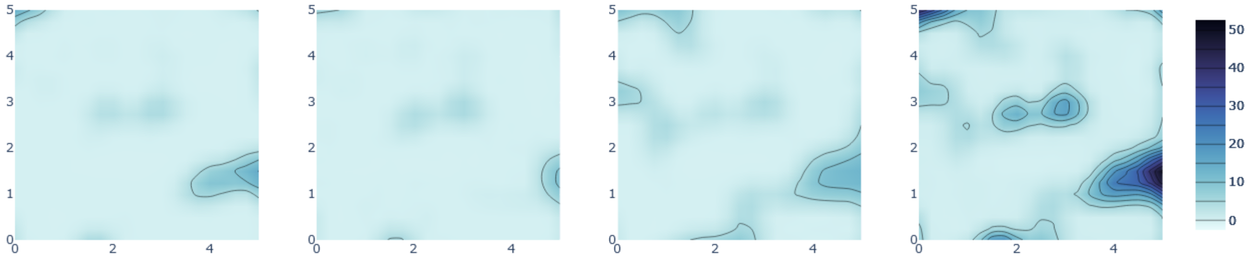


Figure 7.14: *Multimodal threat field error covariance vertex values and fused vertex covariance values at iteration $\ell = 13$.*

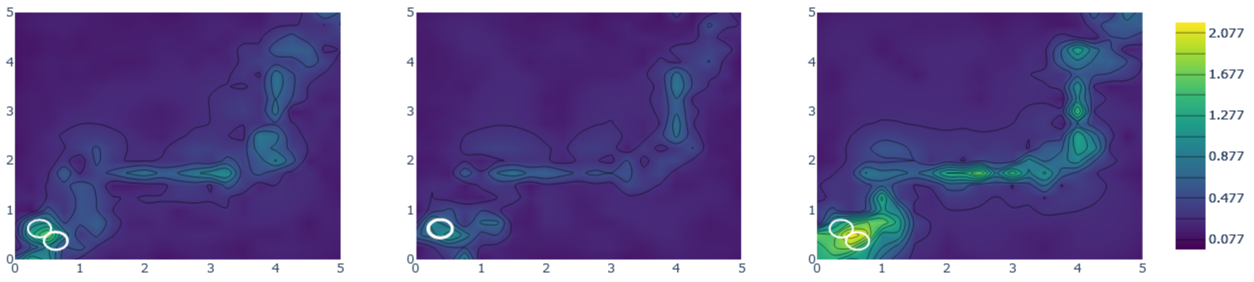


Figure 7.15: *Sensor network configurations overlaying the multimodal SAMI reward values for each threat mode at $\ell = 22$, the final iteration.*

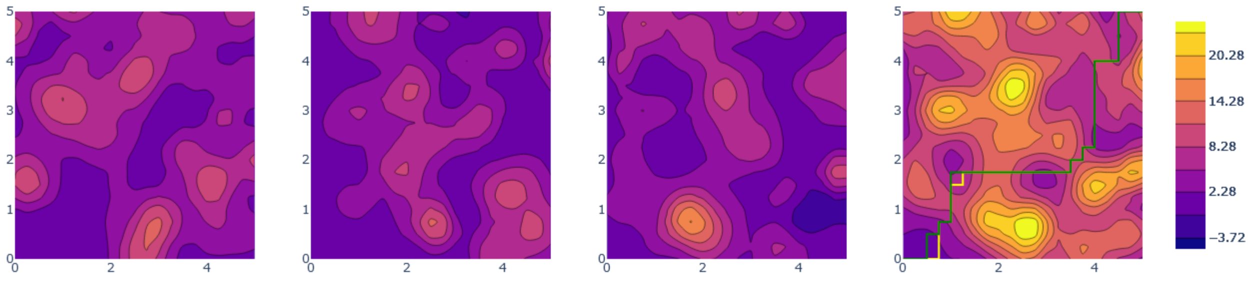


Figure 7.16: *Multimodal threat field estimates and fused threat field estimate (right) along with the estimated optimal path-plan (green) and the true optimal path-plan (yellow) at iteration $\ell = 22$, the final iteration.*

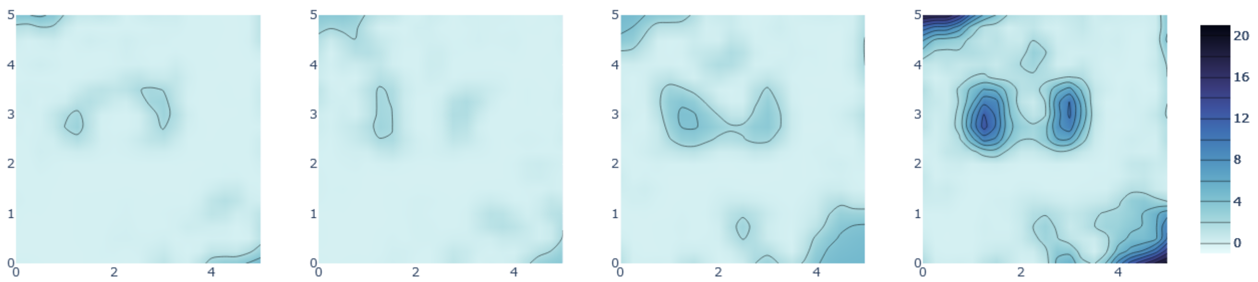


Figure 7.17: Multimodal threat field error covariance vertex values and fused vertex covariance values at iteration $\ell = 22$.

Chapter 8

Active Coupled Sensor Configuration and Path-Planning

8.1 Problem Motivation

In previous chapters, we demonstrated the coupled sensor configuration and path-planning method which finds a path-plan that satisfies prespecified optimality guarantees in minimal iterations and sensor observations. We showed the ability for the algorithm to find a path that converges to a path cost variance optimality threshold in environments which were static and where the acting agent was able to wait for the CSCP algorithm to conclude it's search. This *passive* waiting by the agent works well when waiting does not incur a cost over time. However, in many applications such as radiation exposure, waiting until the algorithm reaches convergence can be costly. Thus, there exists a need for an agent to traverse an environment during the CSCP algorithm operation. In what follows, we detail *active* CSCP for situations in which the acting agent cannot passively wait for the algorithm to converge, but still operates over a static threat environment.

8.2 Active CSCP

We present a realization of the CSCP algorithm for scenarios in which the acting agent cannot passively wait for path-plan optimality convergence due to threat exposure over time. This active CSCP (ACT-CSCP) operates over a static threat environment as in previous chapters, but with the added complexity of a dynamic agent. Herein, the notion of actor horizon and updates to the planning stage are explained. Then, we present a modification to the SAMI surrogate optimization function and numerically show the benefits of this modification for the ACT-CSCP algorithm. We also show a comparison of *waiting* within a safe space in the actor's horizon versus *going* along the current estimated optimal path-plan. We illustrate the changes required to make the CSCP

Active Coupled Sensor Configuration and Path-Planning

```
1: Let $\ell := 0, \boldsymbol{f}_0 := \mathbf{0}, \boldsymbol{P}_0 := \chi \boldsymbol{I}, v_\ell := v_0$
2: Solve for $\boldsymbol{\pi}_0^* := \arg\min \overline{\mathcal{J}}_0(\boldsymbol{\pi})$, from $v_\ell$ to $v_{\mathrm{L}}$
3: while $\mathrm{Var}_\ell(\boldsymbol{\pi}_\ell^*) > \varepsilon$ or $v_\ell \neq v_{\mathrm{L}}$ do
4:     Optimize Sensor Configuration $\mathcal{C}_\ell^*$
5:     Record new measurements $\boldsymbol{z}$
6:     $v_\ell = \boldsymbol{\pi}_\ell^*[\Delta v]$ when $\ell \neq 0$
7:     Increment iteration counter $\ell := \ell + 1$
8:     Find GPR-based threat field estimate $\boldsymbol{f}_\ell$ and error covariance $\boldsymbol{P}_\ell$
9:     Find $\boldsymbol{\pi}_\ell^* := \arg\min \overline{\mathcal{J}}_\ell(\boldsymbol{\pi})$, from $v_\ell$ to $v_{\mathrm{L}}$
```

Figure 8.1: Pseudocode for the Active CSCP algorithm.

algorithm 'active' in Algorithm 8.1.

8.2.1 Actor Horizon and Planning

In this section, we detail the notion of the actor horizon and the changes required for ACT-CSCP in regards to the path-planning efforts. We denote by $\Delta v \in \mathbb{N}$ the number of traversable steps that an agent performs each iteration. The agent's horizon is taken as the set of vertices $\mathcal{H} \subset V$ such that any element in $\mathcal{H}$ is maximum Δv edges E from the current actor location v_ℓ. The value of Δv is proportional to the cycle time of the CSCP algorithm. This cycle time is the agent travel time per iteration $\Delta t_a \in \mathbb{N}$, which is equal to the cumulative time steps required for sensor configuration, field estimation, and planning $\Delta t_a = \Delta t_{sc} + \Delta t_{est} + \Delta t_{pp}$. We assume that $\Delta t_{sc} = n_{sc}\Delta t, \Delta t_{est} = n_{est}\Delta t$ and $\Delta t_{pp} = n_{pp}\Delta t$ where $n_{sc}, n_{est}, n_{pp} \in \mathbb{N}$ are problem specific constants and Δt is the discrete minimum time step. In Figure 8.2, we show the stages for sensor configuration, where the the grey boxes $1, 2, \ldots, N$ represent the sequential optimization for $N = N_{\mathrm{s}}$ sensors and the diagonal boxes represent physical sensor configuration time. Figure 8.3 shows the overall timing diagram. In the diagram example, we say that the agent is capable of making three movements within the discrete cycle time.

To begin the algorithm, we set the agent's current iteration positional vertex as $v_\ell := v_0$. After initialization, we proceed to find the estimated optimal path-plan as the sequence of 4-way connected vertices which minimize $\boldsymbol{\pi}_0^* := \arg\min \overline{\mathcal{J}}_0(\boldsymbol{\pi})$ from the current agent vertex v_ℓ to the

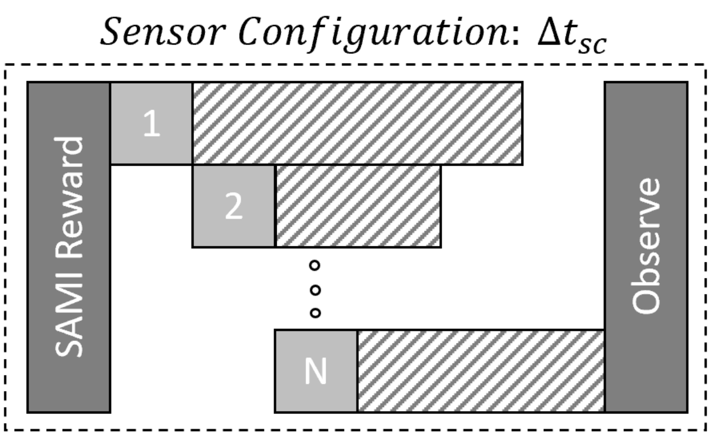


Figure 8.2: Timing diagram for the sensor configuration elements.

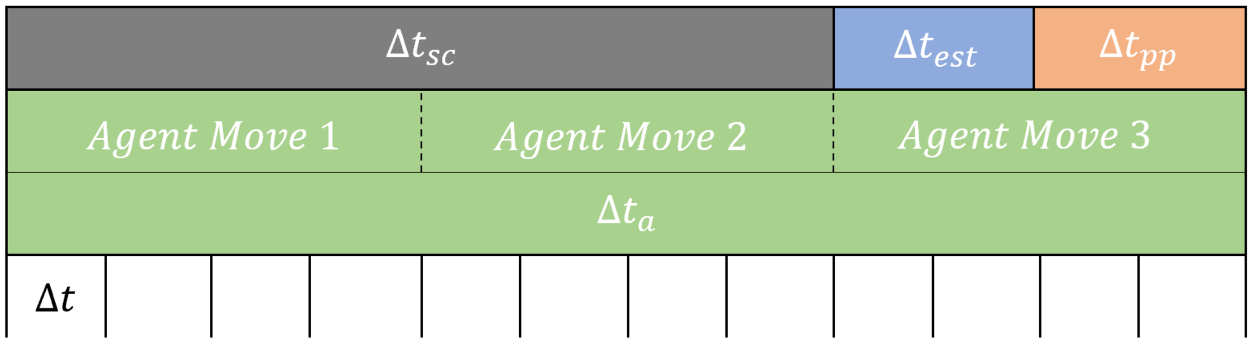


Figure 8.3: Timing diagram for elements of the ACT-CSCP algorithm.

terminal state vertex v_{L}. In what follows, we note that the translation cost is set such that $\Delta t = \Delta p$ and the cost function is updated to reflect this as $\mathcal{J}(\boldsymbol{\pi}) := \Delta t \sum_{j=1}^{\lambda} c(\boldsymbol{p}_{\boldsymbol{\pi}_j})$.

At each iteration $\ell = 0, 1, \ldots, L$, the path-plan for the agent is recomputed. We note that the path-plan differs from previous chapters as it is always taken as the path sequence from v_ℓ to v_{L} rather than v_0 to v_{L}. We also update the current agent vertex to be $v_\ell = \boldsymbol{\pi}_\ell^*[\Delta v]$, meaning that after the sensor configuration stage, sensor observation, and field estimation, the agent is presumed to have achieved the Δv vertex given the prior estimated optimal path-plan.

8.2.2 Environment Estimation and Sensor Configuration

We note in this section that due to the static threat environment assumption, there is no dependency on modeling the effects of time in our statistical field estimate. As such, we may utilize the same Gaussian Process method in Section 2.2.3 for unimodal situations or the Gaussian Process method in Section 7.4 for multimodal environments. Likewise, we note that the sensor configuration procedure follows that of previous chapters for their respective scenarios, but we note that we present a modification to the SAMI surrogate function as detailed in the next section. Additionally, the sensor configuration step utilizes both the region of interest $\mathcal{R}$ and the actor's horizon $\mathcal{H}$.

8.2.3 The Active SAMI Surrogate Function

In Chapter 6 we presented the self-adaptive mutual information surrogate function as written in Equation 6.8 as an efficient replacement for optimizing the TDIG metric directly. It utilized a self-adaptive parameter α which balanced the exploration-exploitation trade-off for sensor configuration using relative entropy as written in Equation 8.6. In Chapter 6 we numerically demonstrated that the adaptive coefficient, which establishes the reward for exploring or exploiting the region of interest, is superior to any fixed coefficient. In what follows, we describe a modification which balances not only exploration or exploitation of the environment, but also the agent's horizon $\mathcal{H}$ to assist during active traversal until the ACT-CSCP algorithm converges to an estimated optimal path-plan or has reached the goal vertex.

Recall in Chapter 6 we defined the reward function for any point $i \in V$ as a mixture of mutual information gain between the region of interest $\mathcal{R}$ and region of interest complement $\mathcal{R}^c$. We amend this definition to include an appropriate weighting for the actor's horizon $\mathcal{H}$, which corresponds to the set of vertices that are Δv vertices from the actor's current vertex v_ℓ. We update the region of interest $\mathcal{R}$ such that $\mathcal{R} := \mathcal{R}_\ell \backslash \mathcal{H}$ and the remaining environment as $\mathcal{R}^c := V \backslash \{\mathcal{R} \cup \mathcal{H}\}$. We note that the entropy and and mutual information of each of these regions for some point $i \in V$ can be obtained with the equations in Chapter 6 with little modification, but we detail in order to be explicit.

The computation of conditional entropy $h(i|\mathcal{H}_{\backslash i})$ can partially be computed as a one-time batch operation and partially computed in parallel for efficiency. For any $i \notin \mathcal{H}$ the vectorized conditional entropy $\boldsymbol{h}(\cdot|\mathcal{H}_{\backslash i})$ is computed as a one-time batch operation:

$$\boldsymbol{h}(\cdot|\mathcal{H}_{\backslash i}) = \frac{1}{2}\ln((2\pi e) diag(\boldsymbol{P}_\ell - \boldsymbol{P}_\ell[\cdot, \mathcal{H}]\boldsymbol{P}_\ell[\mathcal{H}, \mathcal{H}]^{-1}\boldsymbol{P}_\ell[\mathcal{H}, \cdot])) \tag{8.1}$$

For the case where $i \in \mathcal{H}$, we can compute the following conditional entropy equation for any $i \in V$ in parallel batches as:

$$h(i|\mathcal{H}_{\setminus i}) = \frac{1}{2}\ln((2\pi e)(\boldsymbol{P}_\ell[i,i] - \boldsymbol{P}_\ell[\mathcal{H}_{\setminus i}, i]^\intercal \boldsymbol{P}_\ell[\mathcal{H}_{\setminus i}, \mathcal{H}_{\setminus i}]^{-1}\boldsymbol{P}_\ell[\mathcal{H}_{\setminus i}, i])) \tag{8.2}$$

The mutual information I between the $\mathcal{H}$ and $i \in V$ is then calculated as:

$$I(\mathcal{H}_{\setminus i}; i) := h(i) - h(i|\mathcal{H}_{\setminus i}) \tag{8.3}$$

We may then update the SAMI reward function to include the actor horizon set in addition to the region of interest and the complement as follows:

$$\gamma(i) := (1-\alpha)I(\mathcal{R}_{\setminus i}; i) + (1-\beta)I(\mathcal{H}_{\setminus i}; i) + (\alpha+\beta)I(\mathcal{R}^c_{\setminus i}; i) \tag{8.4}$$

Notably, we split the adaptation into two parameters, α and β. Effectively, the parameter α defines the relative mutual information weighting for *non-critical* region of interest vertices. When α is small more reward is emphasized for path-plan discovery. The parameter β defines the relative mutual information weighting for *critical* region of interest vertices and vertices that are within the horizon of the current actor state v_ℓ. We define each as:

$$\alpha := \bar{I}(\mathcal{R}_{\setminus i}; i) / \big(\bar{I}(\mathcal{R}_{\setminus i}; i) + \bar{I}(\mathcal{H}_{\setminus i}; i) + \bar{I}(\mathcal{R}^c_{\setminus i}; i)\big) \tag{8.5}$$

$$\beta := \bar{I}(\mathcal{H}_{\setminus i}; i) / \big(\bar{I}(\mathcal{R}_{\setminus i}; i) + \bar{I}(\mathcal{H}_{\setminus i}; i) + \bar{I}(\mathcal{R}^c_{\setminus i}; i)\big) \tag{8.6}$$

Both parameters are bounded $0 \leqslant \alpha, \beta \leqslant 1$ with $\alpha, \beta \in \mathbb{R}$. We can then obtain $\boldsymbol{\gamma}$ and subsequently Γ and S as in Chapter 6. We note that this does not affect the SAMI cost function Υ. We study the benefits of this new realization of SAMI with dual adaptation parameters in the results section.

$$S(\mathcal{C}_\ell) = \Gamma(\mathcal{C}_\ell) + \Upsilon(\mathcal{C}_\ell). \tag{8.7}$$

We can also say the following about the Active SAMI objective function.

Proposition 5. *The Active SAMI objective function S in Section 8.2.3 is a submodular function.*

Proof. We note that the form of S does not change and that the gain term Γ exhibits the same submodularity as in Proposition 4 regardless of the mixture of mutual information comprising the reward vector, $\boldsymbol{\gamma}$, and as such the proof follows that of Proposition 4, indicating that the Active SAMI function is submodular. $\square$

8.2.4 Algorithm Termination

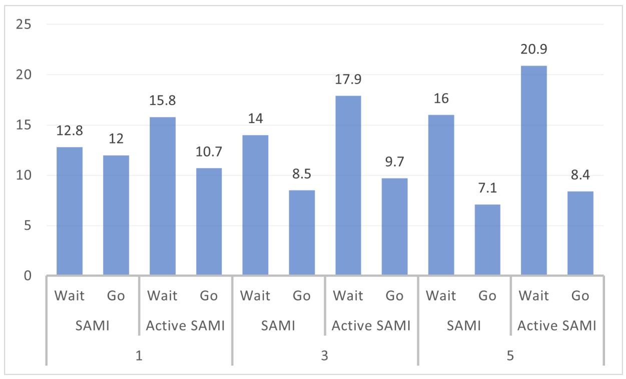
The algorithm terminates if either of two conditions are satisfied. As in previous chapters, we terminate the ACT-CSCP if we converge below the termination threshold ε such that the estimated path cost variance, as formulated in Equation 2.2, is $\mathrm{Var}_\ell(\boldsymbol{\pi}^*_\ell) > \varepsilon$. Alternatively, the algorithm terminates if it achieves the goal state during the iterative process $v_\ell = v_\mathrm{L}$.

8.3 Results and Discussion

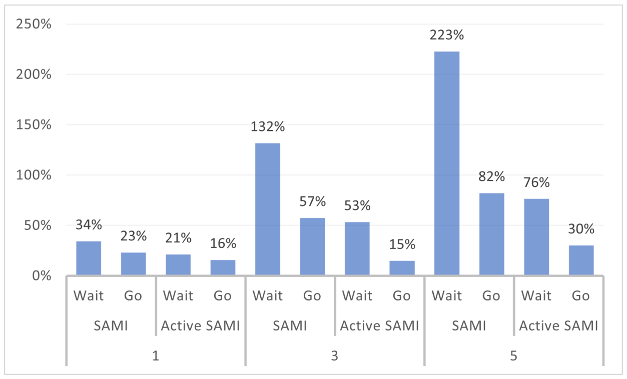
In this section, we aim to address a couple questions in tandem. First, the effect of waiting for path-plan convergence in a locally optimal location versus actively traversing the estimated optimal path-plan given Δv is explored. Waiting is considered by planning the optimal vertex sequence of length Δv within the actor horizon set $\mathcal{H}$ with '5-way' connectivity, meaning that an actor can also remain at the same vertex. Second, a direct comparison of performance between SAMI from Chapter 6 and Active SAMI is explored.

The experiments conducted were unimodal $N_m = 1$ threat fields which were randomly generated as in all previous chapters, with $N_s = 3$ sensors, in an environment of size $25km^2$ and number of vertices in the workspace set to 21^2. The sensors were constrained to $\varrho^{min} = 0.05$ and $\varrho^{max} = 0.5$. We took alternate path-plan samples of count $N_a = 10$. Lastly, the termination threshold was fixed to $\varepsilon = 1$.

The study consisted of performing ACT-CSCP with the SAMI surrogate function and the



(a)



(b)

Figure 8.4: Numerical results for the ACT-CSCP study. (a) Average iterations for each scenario. (b) Average percent suboptimality for each scenario. Studies performed for $\Delta v = \{1, 3, 5\}$.

Active SAMI alternative, which includes the β coefficient for weighting emphasis on exploring the actor's horizon set $\mathcal{H}$. Secondly, each iteration the actor was able to either 'wait' by traversing the '5-way' connected horizon set $\mathcal{H}$, or 'go' by following the estimated optimal path-plan $\boldsymbol{\pi}^*_\ell$. The results of this study are shown in Figure 8.4 and summarized in what follows.

Large horizon reduces iterations at the expense of increased suboptimality

For each 'go' case, we saw that the increase in value of Δv led to a reduction in iterations required to achieve or plan for the goal state. However, this was proportional to increased percent suboptimality. This intuitively indicates that a practitioner can choose between the trade-off of convergence time and threat avoidance performance when setting the value of Δv if it is possible to do so in the application.

Waiting only advantageous under strong assumptions

With no exception, waiting took more iterations on average and incurred more threat exposure percent suboptimality on average than traversing the estimated optimal path-plan. We note that waiting for convergence is beneficial if and only if the waiting position happens to be low cost. Otherwise, a locally minimum basin of threat exposure, when drawn out over many iterations while waiting, can incur more penalties than the 'go' option. Waiting is only justifiable if the immediate vicinity is assumed low-cost given the problem context.

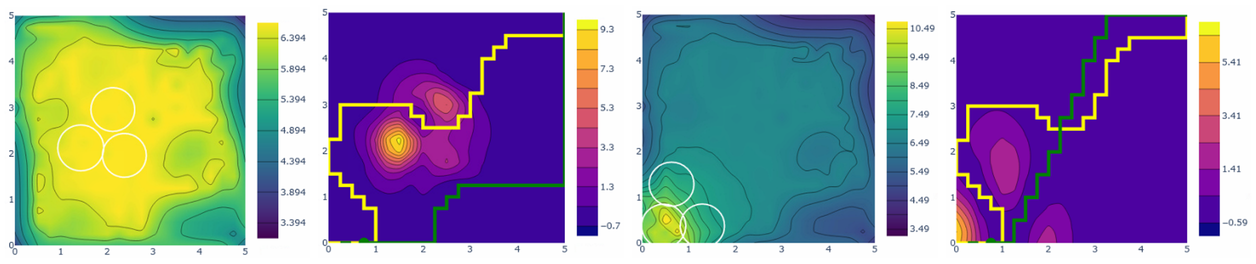
Active SAMI reduces percent suboptimality in dynamic actor scenarios

The key realization of this study is that in all cases, the use of Active SAMI (for go or wait scenarios) improved the percent suboptimality of the path-planning. By introducing consideration of the actor's horizon $\mathcal{H}$, occasional emphasis is placed on ensuring that the active agent is knowledgeable about the threats within it's current iteration horizon. This enables avoidance until the ACT-CSCP algorithm can converge to a near optimal path-plan from v_ℓ to v_{L}.

8.4 SAMI vs Active SAMI Demonstrative Example

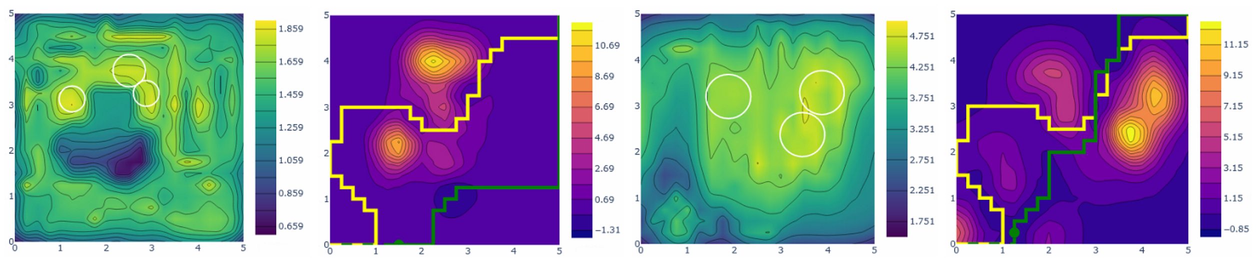
For brevity, we utilize the same environment and scenario setup as in the numerical study. However, in this example, we do not consider waiting as previously presented and just demonstrate the scenario where the actor traverses over Δv vertices in the estimated path-plan. The optimal path-plan cost for this example is 15.08. ACT-CSCP with SAMI took 8 iterations to converge and had path cost of 17.09. Meanwhile, ACT-CSCP with Active SAMI took 10 iterations but had a path cost of only 15.14.

Figures 8.5 through 8.12 show the ACT-CSCP algorithm for both SAMI and Active SAMI reward functions and the corresponding sensor configurations and resulting path-plans and actor movements (dashed green line). In the initial iteration Figure 8.5, the distinction between the two approaches is clear. The SAMI function performs a more exploratory initial sensor configuration whereas using Active SAMI leads to sensor configuration nearby the actor's starting horizon. While both methods do not follow the true optimal path-plan exactly, we see that by iteration $\ell = 4$, Active SAMI is deciding between going vertically or to the right, while SAMI is still looking to the left side of the workspace. This example serves to show that the SAMI surrogate function aims to simply find a near-optimal path plan through various exploration and exploitation sensor configurations. Likewise, after iteration $\ell = 2$, the SAMI based agent traversal backtracks, showing it prone to overshoot. In scenarios in which waiting for such a decision are not possible, the Active SAMI surrogate function places some emphasis on avoidance within the actor horizon while still doing the exploration and exploitation adaptively as in the base version of SAMI.



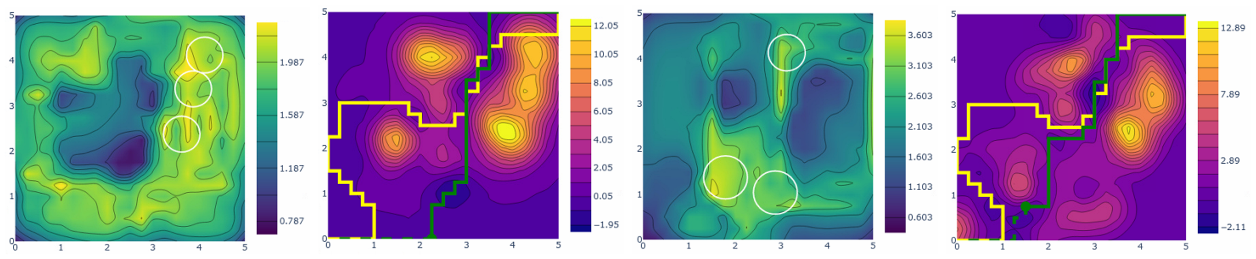
(a) (b) (c) (d)

Figure 8.5: (a) Sensor Configuration with SAMI, (b) SAMI Path-Plan Update, (c) Sensor Configuration with Active SAMI, (d) Active SAMI Path-Plan Update at $\ell = 0$.



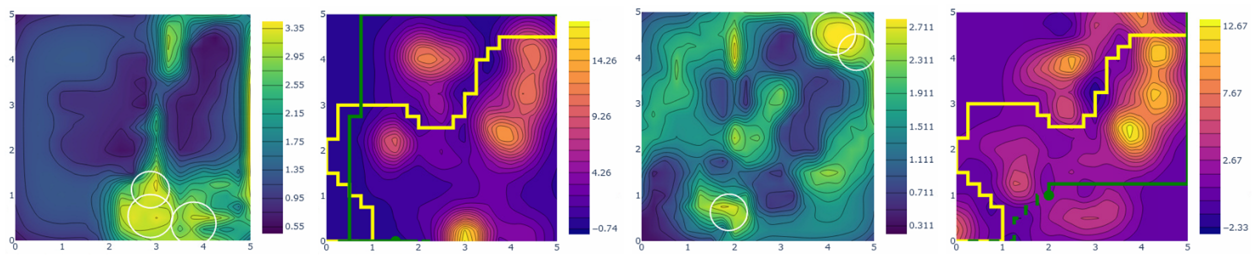
(a) (b) (c) (d)

Figure 8.6: (a) Sensor Configuration with SAMI, (b) SAMI Path-Plan Update, (c) Sensor Configuration with Active SAMI, (d) Active SAMI Path-Plan Update at $\ell = 1$.



(a) (b) (c) (d)

Figure 8.7: (a) Sensor Configuration with SAMI, (b) SAMI Path-Plan Update, (c) Sensor Configuration with Active SAMI, (d) Active SAMI Path-Plan Update at $\ell = 2$.



(a) (b) (c) (d)

Figure 8.8: (a) Sensor Configuration with SAMI, (b) SAMI Path-Plan Update, (c) Sensor Configuration with Active SAMI, (d) Active SAMI Path-Plan Update at $\ell = 3$.

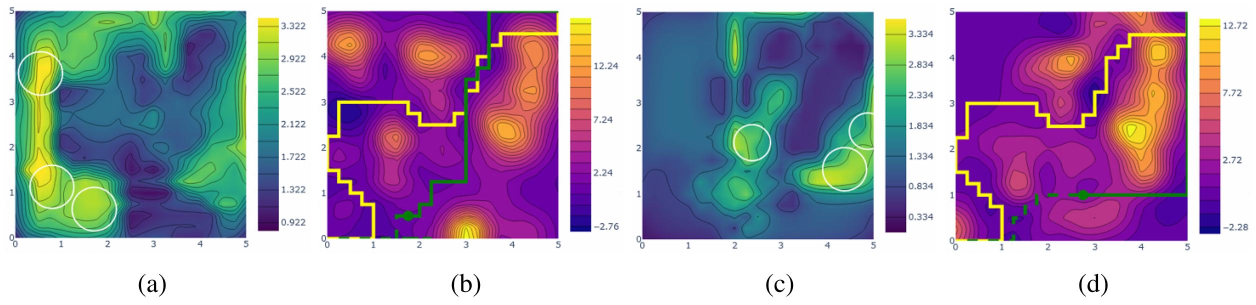


Figure 8.9: (a) Sensor Configuration with SAMI, (b) SAMI Path-Plan Update, (c) Sensor Configuration with Active SAMI, (d) Active SAMI Path-Plan Update at $\ell = 4$.

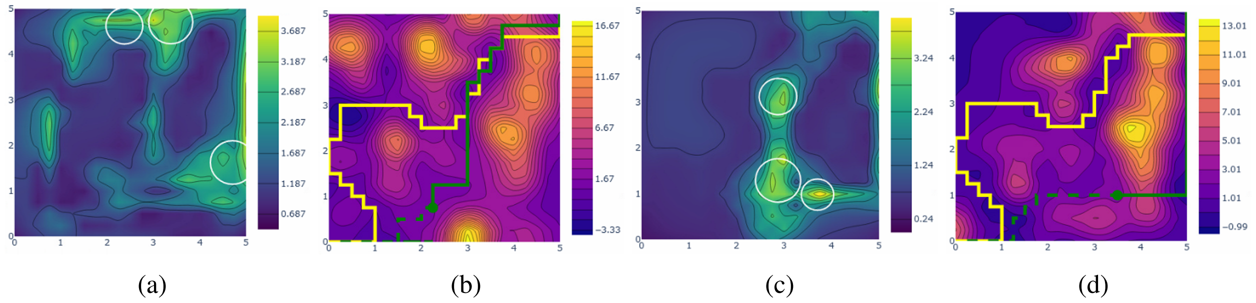


Figure 8.10: (a) Sensor Configuration with SAMI, (b) SAMI Path-Plan Update, (c) Sensor Configuration with Active SAMI, (d) Active SAMI Path-Plan Update at $\ell = 5$.

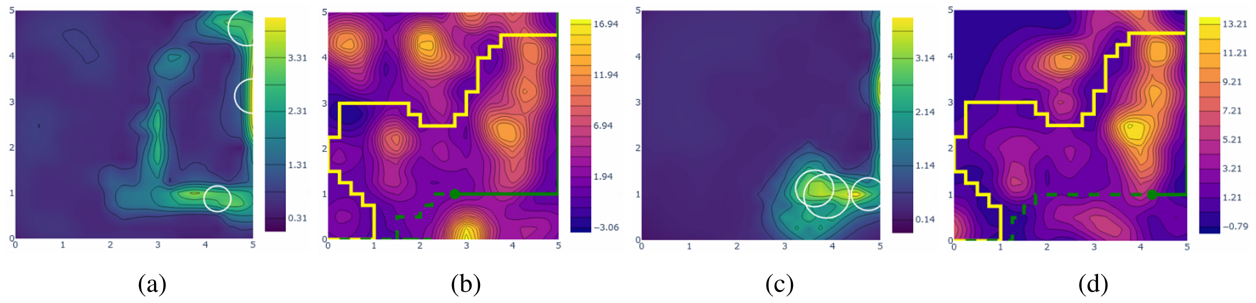


Figure 8.11: (a) Sensor Configuration with SAMI, (b) SAMI Path-Plan Update, (c) Sensor Configuration with Active SAMI, (d) Active SAMI Path-Plan Update at $\ell = 6$.

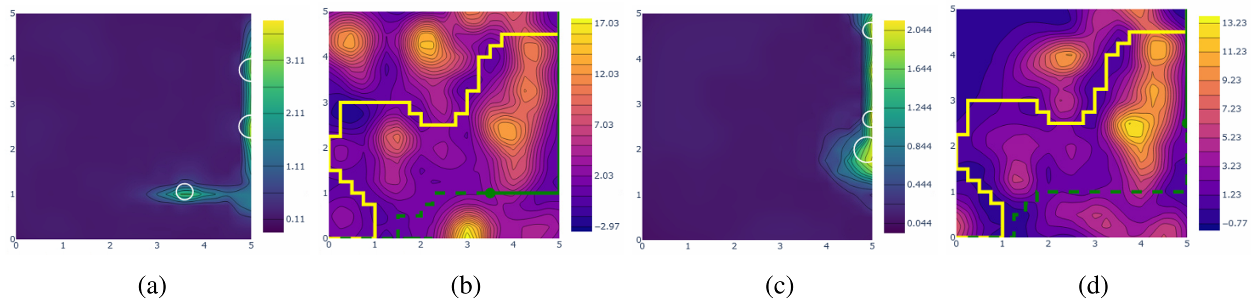


Figure 8.12: (a) Sensor Configuration with SAMI, (b) SAMI Path-Plan Update, (c) Sensor Configuration with Active SAMI, (d) Active SAMI Path-Plan Update at terminal iterations (7 and 9, respectively).

Chapter 9

Conclusions and Future Works

In this dissertation, we provided a *coupled* sensor configuration and path-planning (CSCP) algorithm which allocated sensor network observations of direct consequence to a path-planning task for an agent. We considered sensor network field of view and directly addressed the trade-off in measurement *quality* and *quantity*. The iterative algorithm is shown to converge to a near-optimal path-plan while collecting *minimal* data.

To quantify the performance of coupled sensor configuration and path-planning we developed a *task-driven* information gain (TDIG) metric which measures the reduction in variance of the path-plan. The method models the environment using a nonparametric statistical modeling technique known as Gaussian Process Regression. Inference of this model yields an estimate and covariance of the environment which allows for measures of uncertainty used in the TDIG metric and for determining an estimated optimal path-plan.

For safety-critical scenarios, we developed a Greedy Batch CSCP variation which greedily explores and batches sensor observations to drive path uncertainty below the strict safety-critical optimality requirements. This method successfully finds a near-optimal path-plan within these safety-critical optimality requirements, and does so while reducing runtime and iterations. Likewise, an Exploration Efficient CSCP variation was devised for scenarios in which large amounts of sensors were available. A critical finding with this method is it's capability to expand the performance gap between task-driven and information-driven sensor configuration.

The framework was further expanded to consider qualitative sensor configuration. Through adaptive clustering analysis, fast-approximate sensor configurations were made which closely matched the true optimized sensor configurations. It adaptively switches between exploratory and exploitative clustering and can act as a dimensionality reduction method. The main result is a significant drop in computation time at the expense of a few additional iterations for convergence.

A main result of this dissertation is the ability to perform near-optimal sequential optimization of sensor configuration with a self-adaptive mutual information (SAMI) surrogate function. This function is shown to be submodular and as such the sequential optimization to be $(1 - 1/e)$ near-optimal. We show that the SAMI surrogate, which is used in place of the TDIG metric during sensor configuration optimization, further reduces required iterations to converge and the recurrent function evaluation during optimization is computationally cheaper.

The SAMI method was then applied to scenarios in which the sensor network is comprised of *heterogeneous* sensor payloads observing *multimodal* environments. We demonstrate the extension of SAMI and along with a realization of *mission-level fusion* of the environment utilizing a Gaussian Process kernel for correlated multi-task learning. We detail the various optimization nuances of fully observable, partially observable, and uniquely observable sensor network sensing payloads. We show that unimodal sensor payloads, with their added degree of freedom over partial or fully observable with equivalent sensor counts, are ideal and can be optimized in parallel for faster sensor configuration optimization.

These previous methods operate under the assumption that the agent, for which the path-plan is constructed, is *passive* as it awaits a path-plan after the iterative sensor configuration loop. The CSCP algorithm is updated to account for *active* planning such that at each discrete time step the agent is capable of traversal. The Active SAMI surrogate function is developed to include actor's traversal horizon, enabling sensor configuration for *avoiding* spatially-proximal threats. We show that waiting in passive threat environments is far more suboptimal than active traversal. Likewise, we describe how the active method may be applied to the scenario in which the environment is no longer static, but dynamically varying with time.

The result of this dissertation is an efficient framework for sensor configuration as it pertains to spatial path-planning. A variety of variations are provided to deal with application specific cases, numerical methods for lowering computational complexity, and extensions to heterogeneous sensor networks, active agents and dynamic environments. We show the sensor configuration can be solved sequentially. In unimodal heterogeneous networks sensor configuration can be solved both sequentially and amongst each mode in parallel.

9.1 Future Work Considerations

Herein, future work considerations to extend the outcomes of this dissertation are presented. We describe additionally constraints and parameters that can be considered, objective function considerations, and the ability to actively select which sensors make observations or even choose when and at what level to perform sensor fusion. Finally, an application for real-time sensor coordination in tracking scenarios is described.

9.1.1 Sensor Field of View Constraints and Parameters

In this dissertation, we considered a sensor configuration as the collection of each k^{th} sensor location and radius $\mathcal{C} = \{\boldsymbol{s}_1, \varrho_1, \boldsymbol{s}_2, \ldots, \varrho_{N_\mathrm{s}}\}$. The assumption was threefold, (1) the sensor field of view was parallel to the 2D environment, (2) the field of view noise is uniform and only grows monotonically with increasing radius, and (3) the field of view was area-based and circular. Using an example of a UAV, we may need to consider 3D coordinates and sensors equipped to a gimbal which can rotate about all three axes. The parameter space then becomes: heading, 3D coordinates, and the 3D sensor angle. Each parameter could additionally be constrained to a 3D coordinate space and the 3D angle confined to a subset of a hemisphere. With such a high dimensionality, the SAMI surrogate function for sequential optimization becomes highly beneficial. However, it would need to be modified to account for non-parallel and non-circular field of views. Sensors also do not necessarily need to be area-based. Of interest is studying how noise is effected by elongated field of view away from the sensing apparatus for non-parallel observations.

9.1.2 Sensor Network Costs

The content of this work implicitly assumed that the mobile sensor network was instantly configurable to any location within the workspace without any penalties incurred. However, in UAV applications, there are many costs or heuristics that can be considered during sensor configuration. In a UAV setting, we could consider the lateral, ascent, and descent speeds along with maximum battery lifetime. Additionally, the amount of data obtained from heterogeneous sensor types can vary. All of these parameters could be used to regularize the sensor configuration objective function. For example, we could add a heuristic for the amount of data collected, valuing sensors which collect less data with just as much information quality.

9.1.3 Selective Sensing

In tandem with the suggestion of adding sensor network costs, we could extend the sequential optimization to the concept of selective sensing. Utilizing the SAMI objective function, which is submodular and exhibits a diminishing returns property, we could prematurely stop finding sensor configurations for all available sensors once the returns from optimizing each sensor in sequence falls below a specified threshold. Such an implementation could further reduce the number of required observations of the environment in achieving the near-optimal path-plan and save time during sensor configuration optimization.

9.1.4 Human-in-the-loop Considerations

Early adoptions of the CSCP algorithm for real-world physical systems may not involve *full* autonomy of the sensor networks and agent planning. As such, there may be need for human intervention to help support and guide the CSCP efforts and aid in path-plan convergence. A user interface (UI) could be developed which allows for a practitioner to intervene during the CSCP iterative procedure. A few ideas for how this could benefit the CSCP methodology are as follows.

Human intervention could provide support for early termination of the CSCP algorithm prior to discovering a path-plan which satisfies the path cost variance is below a specified termination threshold. A practitioner could visually confirm that the path is satisfactory and therefore reduce the number of iterations for finding a path-plan. Additionally, a human could select locations for the agent to travel during active CSCP scenarios as in Chapter 8.

Human intervention could also be applied to sensor network configuration tasks. After determining optimal sensor configurations, a user could specify which sensors in the network will be sent to these configurations and optionally only select a certain amount to be deployed to these configurations. Additionally, given a UI for the environment, a user could circle or prune regions

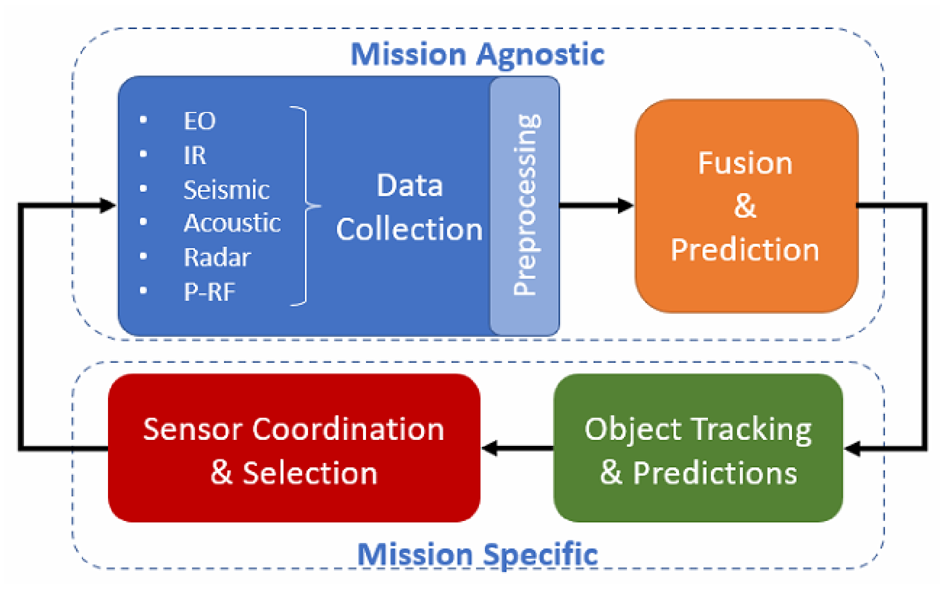


Figure 9.1: Multimodal Coupled Sensor Configuration and Tracking Loop

of the environment for selecting a custom region of interest separate from what is described in the previous chapters. This could allow sensor configuration to optimize over user defined regions and reduce computational complexity if the workspace resolution is large and/or the planning environment is a very large area.

9.1.5 Efficient Real-Time Sensor Configuration for Tracking

Consider a scenario where various vehicle types with different features such as a van, a car, a truck, a diesel 4-wheeler, and a gas 4-wheeler need to be tracked. The vehicles may have difference colors, emit sound and ground vibrations at different frequencies, and have variable thermal readings. Given a sensor network comprised of heterogeneous sensors such as electro-optical (EO), infrared (IR), acoustic arrays, passive-RF (P-RF), sesimometers, and radar. It would then be the task of this collection of sensors to classify and track the various vehicle types. An example multimodal coupled sensor configuration and tracking loop could look something like Figure 9.1. Mission agnostic tasks would be the data collection and processing of the various sensor types and the fusion and prediction thereof. Mission specific tasks would involve tracking the identified objects, predicting their movements, and coordinating select sensors for data collection.

One of the main challenges with this multimodal tracking problem is that not all sensing

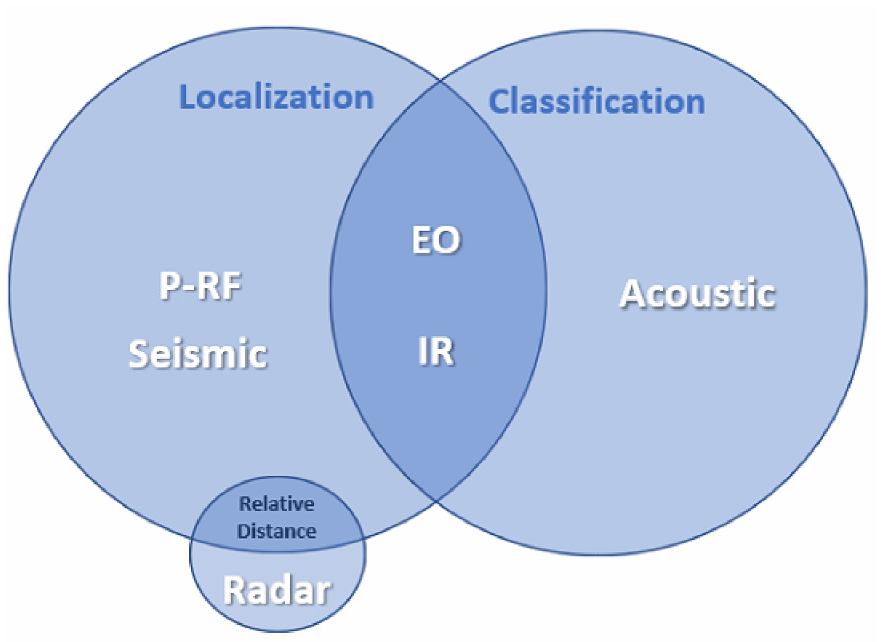
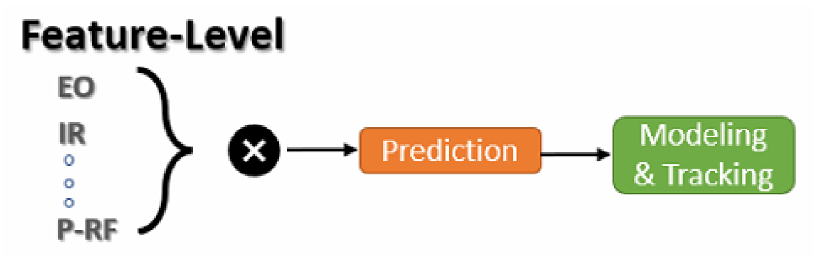


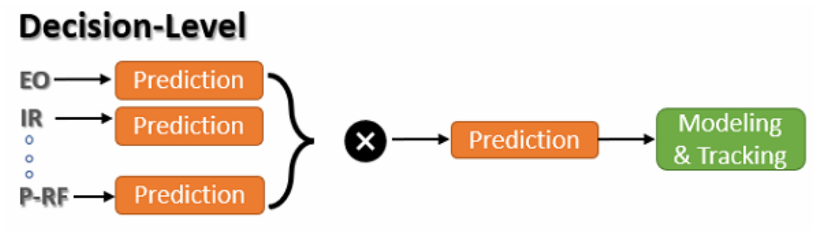
Figure 9.2: Various sensors and their applicability mapping to the vehicle tracking problem

modalities observe the same type of mission specific data. A diagram of the types of data that can be extrapolated for such a problem is shown in Figure 9.2. To make the problem even more challenging, the fusion and prediction stage could also be selected as a meta-sensor type, i.e. a unimodal observation prediction from EO or a fused multimodal prediction from EO and IR. Specifically, selections could be made for feature level, decision-level, or mission-level fusion as shown in Figure 9.3, which would act as meta-sensors for a realization of CSCP. Criteria for selection could include the aforementioned topics in the Future Works section, namely the amount of data collected, the physical costs, information quality, etc. Additionally, considerations about the type of information (classification and/or localization) need to be accounted for during sensor configuration.

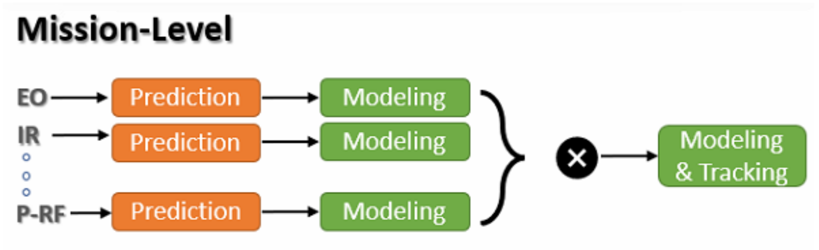
CSCP could be modified such that the threats are modeled as the likelihood of an object at a particular location and the path-plan the projected routes resulting from a tracking algorithm. The region of interest would then be composed of the projected future vehicle tracks and optionally sampled from the statistical model. The objective could be changed to maximizing the accuracy of vehicle identification and tracking over a finite time-span of interest.



(a)



(b)



(c)

Figure 9.3: Feature-level fusion (a), decision-level fusion (b), and mission-level fusion (c) for the tracking problem.

Appendix A

CSCP for Multi-Agent Multi-Goal Scenarios

We make the following application note of the CSCP algorithm applied to scenarios which require multiple-agents which need to arrive at the same goal location, but which start at a variety of initial locations. Alternatively, it could also be applied to applications which require a many-to-many agent/goal location scenario. In this scenario, each agent has a unique goal vertex state to achieve. Luckily, the CSCP algorithm can work for such scenarios with the following modifications to key elements of the algorithm.

First, consider that there are maximum $N_{aa} \in \mathbb{N}$ agent/goal pairings which need to be routed in an unknown threat environment. Therefore, the initial path-plan estimate and subsequent path-plan estimates need to be found as $\boldsymbol{\pi}_\ell^{(i)*} \, \forall \, i \in N_{aa}$ for all iterations $\ell \in \{L\}$. The region of interest is naturally updated to the union of each agent's path-plan individual region of interest as:

$$\mathcal{R}_\ell := \mathcal{R}_\ell^{(1)} \cup \mathcal{R}_\ell^{(2)} \cup \cdots \cup \mathcal{R}_\ell^{(N_{aa})}$$

From this formulation of $\mathcal{R}_\ell$, the region of interest incidence vector $\boldsymbol{r}$ can be determined for use in the TDIG metric or $\mathcal{R}$ for the SAMI surrogate. We note that alternate statistically feasible path-plans can be sampled as introduced in Chapter 4. Finally, the termination condition can be computed as:

$$\sum_{i=1}^{N_{aa}} \mathrm{Var}_\ell(\boldsymbol{\pi}_\ell^{(i)*}) > \varepsilon \tag{A.1}$$

The termination check of Equation A.1 is an additive function of the individual path-cost variance values for each agent/goal route. Due to this additive nature, the problem can be decoupled and independently solved. This separability ensures the aforementioned properties of the CSCP algorithm hold in multi-agent multi-goal scenarios.

Appendix B

CSCP in Spatiotemporally Evolving Threat Fields

There exists many use cases in which the threat environment is not static, but dynamically evolving through time. In this situation, we require *time-varying* coupled sensor configuration and path-planning to handle spatiotemporal threats. Given a spatiotemporally evolving threat field setting, it may be necessary for an agent to *avoid* threats while simultaneously trying to create a path-plan which assists the agent in *achieving* it's goal state. We directly build upon the sequential sensor configuration abilities demonstrated in Chapter 6 and the active CSCP formulations of Chapter 8. We detail necessary and suggested modifications to the statistical field estimate which handles spatiotemporally evolving threats, allowing for forward looking environment predictions.

B.1 TV-CSCP

In what follows, we present an application of coupled sensor configuration and path-planning for time-varying threat fields (TV-CSCP). We note that the algorithm follows that of ACT-CSCP from Chapter 8, but has special sensor configuration conditions. We modify the threat estimation and planning to handle spatiotemporal threats, where the threat field is now represented as $c(\boldsymbol{x}, t) \to \mathbb{R}_{>0}$. The termination condition is shown to conditionally converge when provided a finite mission time. We note that the following framework modifications follow the same procedure of Algorithm 8.1 but with nuances.

B.1.1 Problem Overview

We make the following modifications to the problem formulations from Chapter 2. First, consider a finite mission-time $N_T \in \mathbb{N}$. We may then enforce that the terminal iteration is $L \leqslant N_T$ and additionally is taken such that it is larger than or equal to the minimum realizable path-length

$N_T \geqslant \lambda$ for a given set of start and goal vertices v_0 and v_{L}. For a 4 or 5-way connected workspace we make the following assumption about the mission-time.

Assumption 3. *A finite mission-time $N_T \in \mathbb{N}$ is such that $N_T \geqslant 2\sqrt{N_{\mathrm{g}}} + 1$.*

This assumption assures that a valid path is achievable within the allotment of mission-time N_T. The finite mission-time assures we can utilize Djikstra's algorithm for finding a minimal cost path, as it confines the search space to a 3D workspace with the added dimensionality of time. Likewise, we can find an optimal path-plan $\boldsymbol{\pi}^*$ as a minimum-cost path that minimizes the time-based function $\mathcal{J}(\boldsymbol{\pi}) := \Delta p \sum_{k \in \lambda} c(\boldsymbol{p}_{\boldsymbol{\pi}_k}, k\Delta t)$. Relevant incidence vectors, such as the path incidence vector, can be expanded to accommodate the mission-time as $\boldsymbol{v}_{\boldsymbol{\pi}} \in \{0,1\}^{N_g N_T}$.

B.1.2 Time-Dependent Threat Estimation

Until now, the kernels considered for the Gaussian Process modeling were stationary, that is, they only depended upon the relationship between pairs of parameter inputs. Instead, we may utilize *non-stationary* kernels which produce outputs which are not spatially agnostic. We recommend the reader review the work of (Duvenaud et al.), which details the various kernel structure properties of the sum and product of various stationary and non-stationary kernels. Given the context of the environment being modeled, it may be beneficial to learn non-stationary linear trends, periodicity, etc. Some common kernels for capturing time-based trends (linear, polynomial, periodic) are written as follows.

$$\boldsymbol{K}^{LIN}[i,j] := \theta_c \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j \tag{B.1}$$

$$\boldsymbol{K}^{POLY}[i,j] := (\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j + \theta_c)^d \tag{B.2}$$

$$\boldsymbol{K}^{PER}[i,j] := \exp\left(-2 \sum_k \sin^2\left(\frac{\pi}{\theta_p}(\boldsymbol{x}_{i,k} - \boldsymbol{x}_{j,k})\right)/\theta_\ell\right) \tag{B.3}$$

These kernels could then be applied directly to the existing kernel structures from previous chapters via their summation or multiplication.

B.1.3 Sensor Configuration and Algorithm Termination

The methodology for optimizing sensor configuration largely remains the same, as we utilize the SAMI formulation as in Chapter 8 which was shown to be advantageous in scenarios where the agent actively traversed the threat environment. Likewise, the Active SAMI surrogate function emphasizes information gain within the actor horizon $\mathcal{H}$, which is conducive to avoiding time-varying threats. In regards to algorithm termination, we note that this also does not change, as the GP regression with the ability to learn trends over time, it is capable of finding (though for complex environments difficult to achieve) a path-plan with a specified termination threshold $\varepsilon > 0$. Otherwise, the algorithm proceeds until $v_\ell = v_{\mathrm{L}}$.

Appendix C

Sparsity Techniques for CSCP

This appendix deals techniques for sparsity at various operations within the CSCP algorithm. Herein, we describe a selection of methods which can be used to reduce time complexity for the environment modeling, aggregation of measurement data, and the SAMI function.

C.1 Sparse Environment Modeling

The statistical environment models in this work rely on Gaussian Process Regression techniques for nonparametric estimation of the underlying threat field. While the exact determination of a Gaussian Process model's hyperparameters was presented, the inference time scales unfavorably with the amount of training data that is aggregated when inference is performed $\mathcal{O}(n^3)$, where n is the amount of training data. The following approaches are provided as a non-exhaustive selection of strategies that a practitioner could use to reduce this computational burden that may be helpful in certain applications.

C.1.1 Recursive Updates

One of the more direct approaches to deal with the growing amount of training data is to deal with the modeling strategy directly. Rather than aggregate data each iteration of the CSCP algorithm, one could define and train a GP model for just the current iteration training data. Once the model is trained, we store the current field estimate and estimation error covariance and toss out the previous sensor measurements. Given the assumption that the learned model follows a Gaussian likelihood, a recursive weighted update may be applied, fusing the current and previous model estimation. Let $\boldsymbol{\mu}_*$ and $\boldsymbol{\Sigma}_*$ be the inferred values from the GPR model at iteration ℓ. Using these posteriors, we calculate the threat estimate and error covariance based on the previous estimate and error covariance, as follows:

$$\Lambda := (\boldsymbol{P}_{\ell-1} + \boldsymbol{\Sigma}_*)^{-1} \tag{C.1}$$

$$\boldsymbol{f}_\ell := \boldsymbol{\Sigma}_*\Lambda\boldsymbol{f}_{\ell-1} + \boldsymbol{P}_{\ell-1}\Lambda\boldsymbol{\mu}_*, \tag{C.2}$$

$$\boldsymbol{P}_\ell := \boldsymbol{P}_{\ell-1}\Lambda\boldsymbol{\Sigma}_* \tag{C.3}$$

We provide a note of caution in using this formulation. Due to the spatial properties of certain kernels used in GP regression, the posterior estimates away from each iterations training data may not capture the inherent uncertainty fully. As such, it may be overconfident in these regions and generate a biased estimator of the underlying field model.

C.1.2 Sparse GP

In contrast to generating a new model each iteration, a practitioner could utilize a GP variation called Sparse Gaussian Processes or Sparse Variational Gaussian Processes (and other similar variations) (Titsias, 2009). These methods rely on a set of m inducing points, which are such that $m << n$, where n is the training data. In basic terms, the inducing points are either randomly selected from the training set and optionally optimized to locations within the training set domain. In doing so, the GP model requires only using these m inducing points for inference and thereby reduces the prediction time complexity to $\mathcal{O}(nm^2)$.

C.1.3 Online GP

Of notable consideration is the approach of training the GP in an online sequential fashion as training data is made available. Conceptually, the GP would make modifications with the newly available data while not having to relearn a good structure from scratch. A particular method of note is the WISKI algorithm which utilizes the Woodbury Identity in combination with structured kernel interpolation for constant-time $\mathcal{O}(1)$ online operations (Stanton et al., 2021). The method additionally has caching capabilities of sub-computations. Such a method enables fast inference and training while not requiring keeping a history of previous measurement data.

C.2 Observation Sparsity

Another area in which sparsity may be enforced is by directly altering the measurements which are collected by the sensor network. Sometimes, too much data can be a burden even when applied to the aforementioned sparse modeling approaches. Thus, we require a way to either combine the provided information or reject some of that information depending on it's quality.

C.2.1 Pooling to Workspace Vertices

One potential method for reducing the number of measurements aggregated prior to fitting the statistical environment model is by pooling data which is spatially relevant to the workspace vertices. Figure C.1 depicts two such methods, K-Nearest Neighbors (KNN) and pooling within a cell. KNN operates by finding k measurements which are closest to a particular workspace vertex and combines just those points. By contrast, the cell combines all points which are contained in that area. KNN can therefore have a smoothing effect while the cell-based approach is a holistic approach to combine the data within a defined spatial region in which a sensor FoV cover $\mathcal{F}$ intersects some point $i \in V$. The observations are combined in a weighted manner at any vertex $i \in V$ for any $j \in \mathcal{W}$ such that j corresponds to the $M \in \mathbb{N}$ candidate observations for pooling:

$$\boldsymbol{z}_i = (\boldsymbol{z}_j^{\intercal}\boldsymbol{\sigma_j^2})/(\mathbf{1}^{\intercal}\boldsymbol{\sigma_j}^2) \tag{C.4}$$

$$\sigma_i^2 = (\Pi_{j=1}^{M}\boldsymbol{\sigma_j^2})/(\mathbf{1}^{\intercal}\boldsymbol{\sigma_j}^2) \tag{C.5}$$

C.2.2 Novelty Criteria and Surprise

Rather than pooling data or otherwise combining it, methods for statistically pruning data which is not considered novel or does not possess a notion of surprise to the existing model is considered. Online learning using a Bayesian surprise metric is developed in (Hasanbelliu et al., 2012)

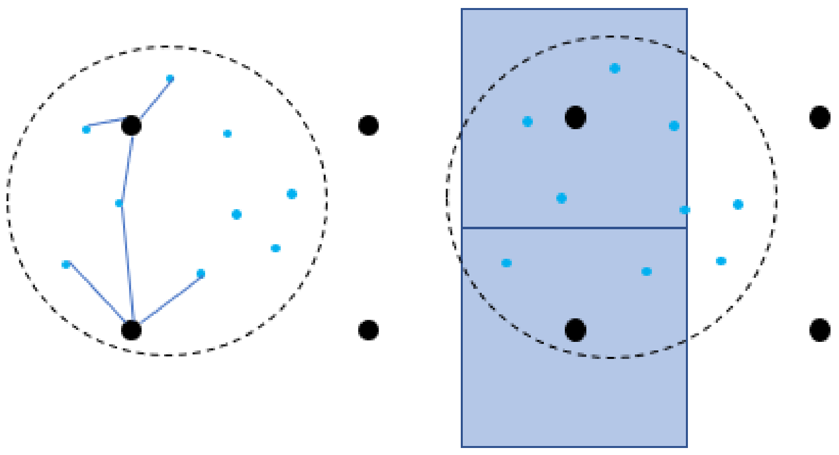


Figure C.1: Example of observation combinations. (a) Combining with K-Nearest Neighbors with $k = 3$. (b) Combining with cell region pooling. In these figures are the sensor Fov (dotted circle), workspace grid points (black dots), observations (blue dots).

by assessing the prior and posterior distributions of the existing model when the data of question is introduced. The data is classified as an outlier (discarded), interesting (kept for updating the model), or redundant (discarded) based on an approximation to the Kullback-Leibler divergence between the model with and without the new data point.

C.3 Sparse SAMI Reward Function

In Chapter 6 we presented the SAMI surrogate function, in Chapter 7 the multimodal SAMI updates, and in Chapter 8 the modified Active SAMI surrogate function. The reward vector Γ requires an upfront computational burden prior to optimizing the SAMI surrogate function for optimal sensor configuration. We noted that for each variation of SAMI the scenario where $i \notin \mathcal{R}$, $i \notin \mathcal{H}$, and $i \notin \mathcal{R}^c$ the conditional entropy for each i could be batch computed given the respective formulation. Likewise, whenever i was in any of these sets, we could compute the entropy for i given each set exclusive of i in parallel. In either case, however, matrix inversion was required.

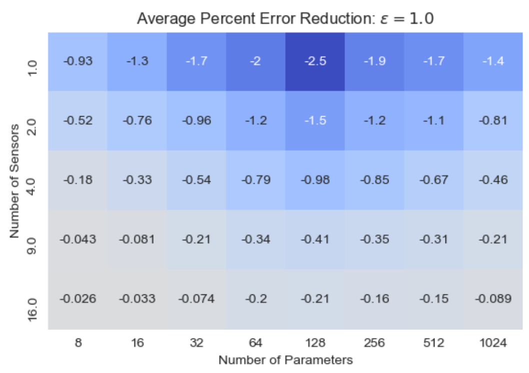
We note that the RBF Kernel $\boldsymbol{K}^R$ has a covariance matrix structure equivalent to the correlation matrix. Namely, for any point $i, j \in \mathbb{N}$ the correlation of $f(i)$ and $f(j)$ where f can be described as a normal distribution is exactly $\kappa(i, j)$. We can then define a region of support

$\mathcal{R}^{sup} \subset V$ as any $i \in V$ that satisfies $\kappa(i, j) \geqslant \varepsilon_{sup}$ for any $j \in \mathcal{R}, \mathcal{H}, \mathcal{R}^c$, respectively, and ε_{sup} is the threshold for minimum considered correlation.

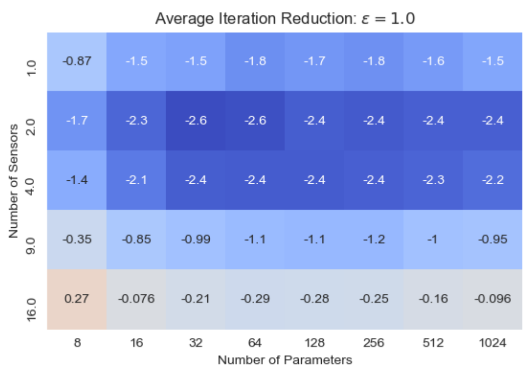
Given the region of support $\mathcal{R}^{sup}$, we can selectively compute a subset of workspace vertices that satisfies the threshold ε_{sup} as batch or parallel operations as previously stated. For any $i \notin \mathcal{R}^{sup}$, we can set the Sami Reward vector as $\boldsymbol{\gamma}[i] = 0$.

Appendix D

Supplementary Chapter 2 Results

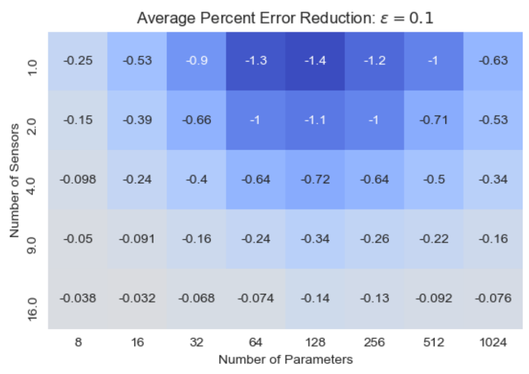


(a)

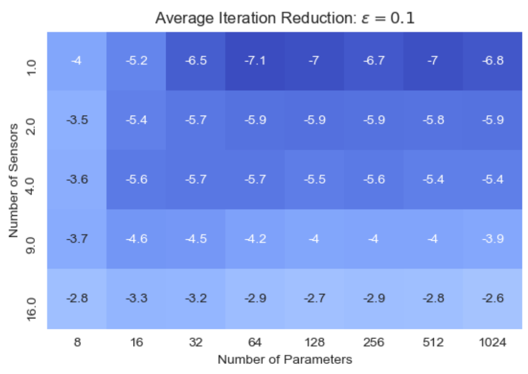


(b)

Figure D.1: Average percent error (a) and iterations (b) reduction by using CSCP instead of Info-Max for various parameters and $\varepsilon = 1$.

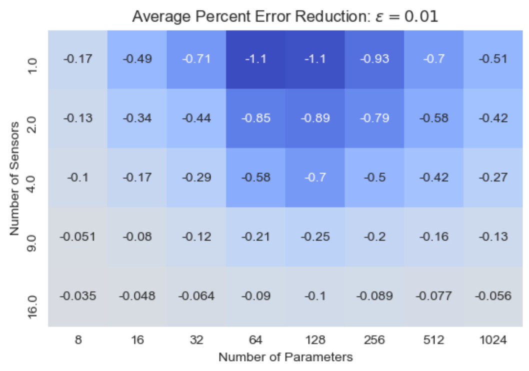


(a)

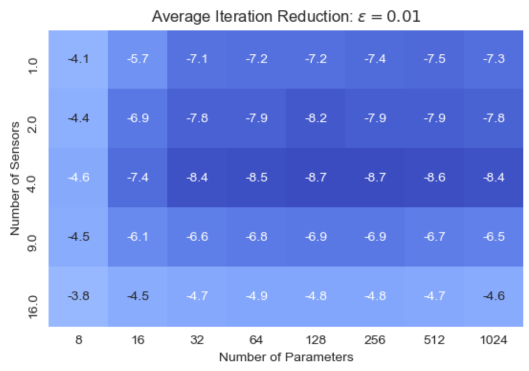


(b)

Figure D.2: Average percent error (a) and iterations (b) reduction by using CSCP instead of Info-Max for various parameters and $\varepsilon = 0.1$.

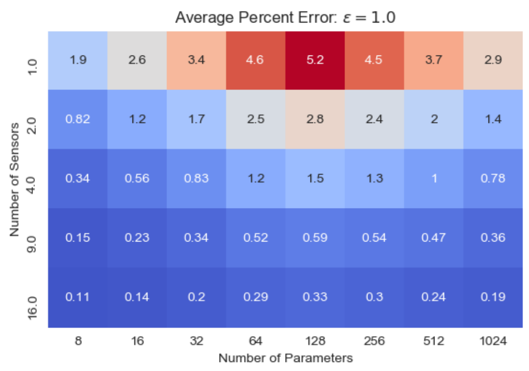


(a)

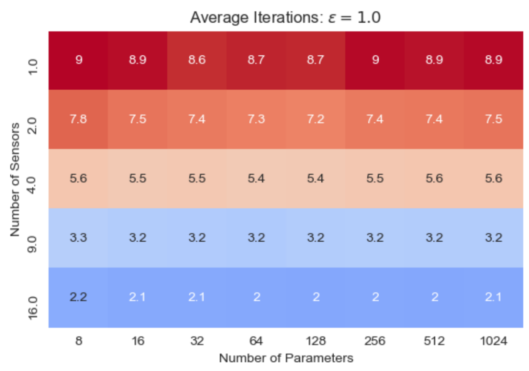


(b)

Figure D.3: Average percent error (a) and iterations (b) reduction by using CSCP instead of Info-Max for various parameters and $\varepsilon = 0.01$.

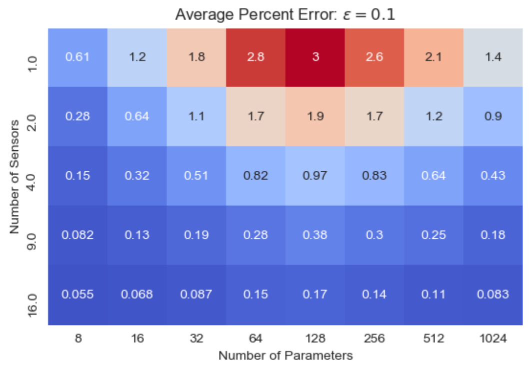


(a)

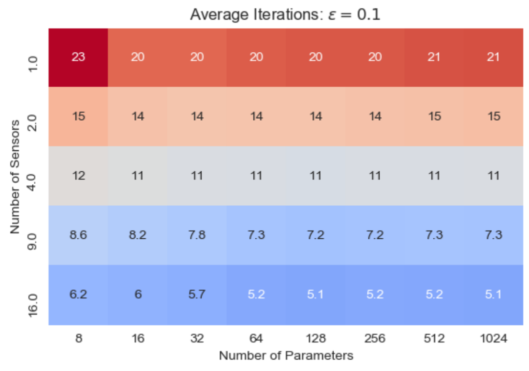


(b)

Figure D.4: Average percent error (a) and iterations (b) for Info-Max for various parameters and $\varepsilon = 1$.

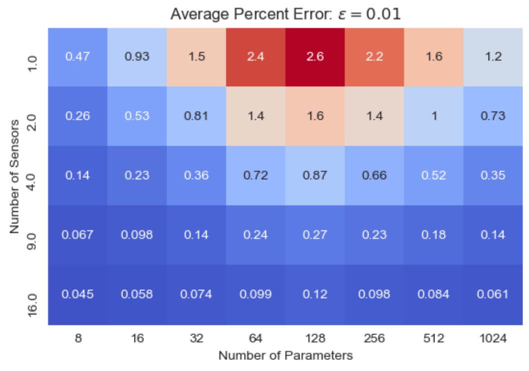


(a)

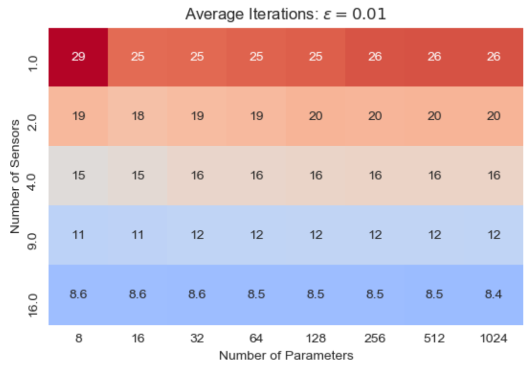


(b)

Figure D.5: Average percent error (a) and iterations (b) for Info-Max for various parameters and $\varepsilon = 0.1$.

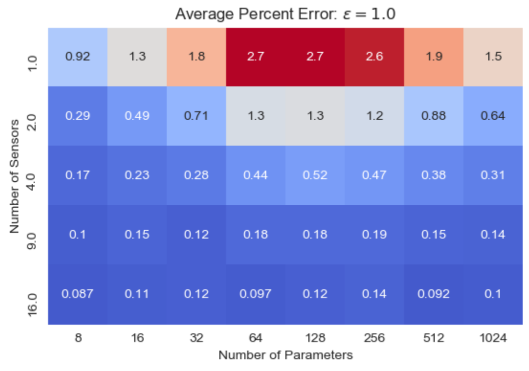


(a)

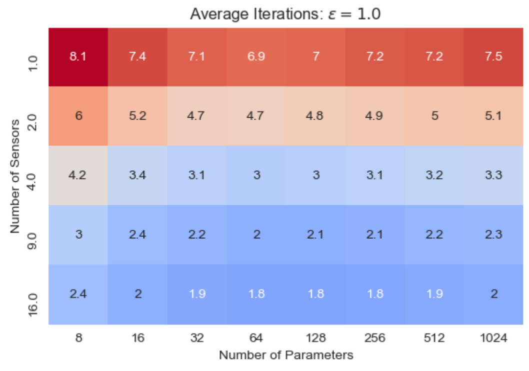


(b)

Figure D.6: Average percent error (a) and iterations (b) for Info-Max for various parameters and $\varepsilon = 0.01$.

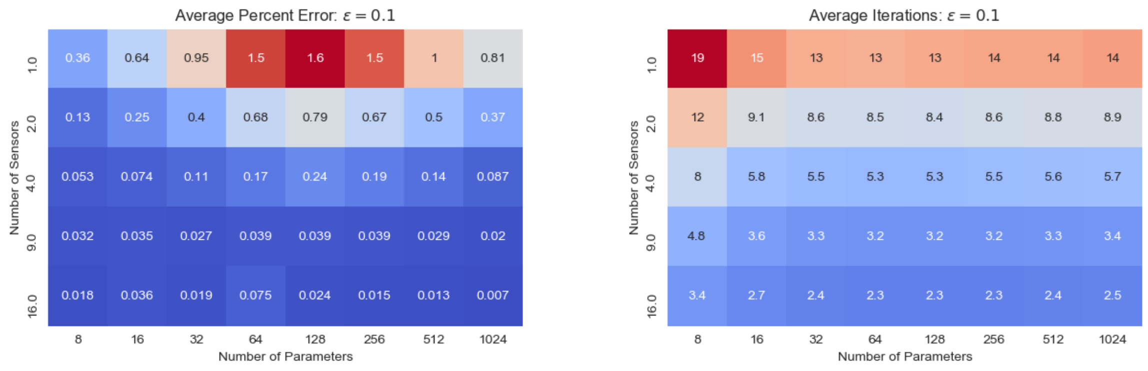


(a)



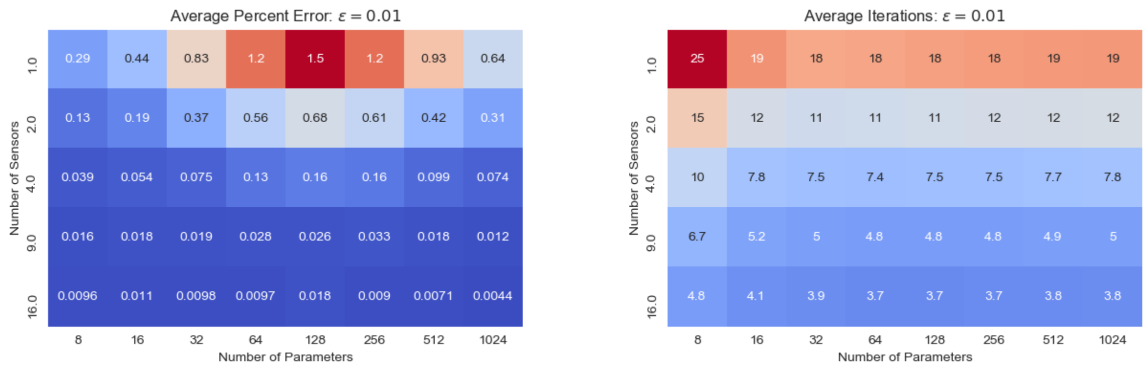
(b)

Figure D.7: Average percent error (a) and iterations (b) for CSCP for various parameters and $\varepsilon = 1$.



(a) (b)

Figure D.8: Average percent error (a) and iterations (b) for CSCP for various parameters and $\varepsilon = 0.1$.



(a) (b)

Figure D.9: Average percent error (a) and iterations (b) for CSCP for various parameters and $\varepsilon = 0.01$.

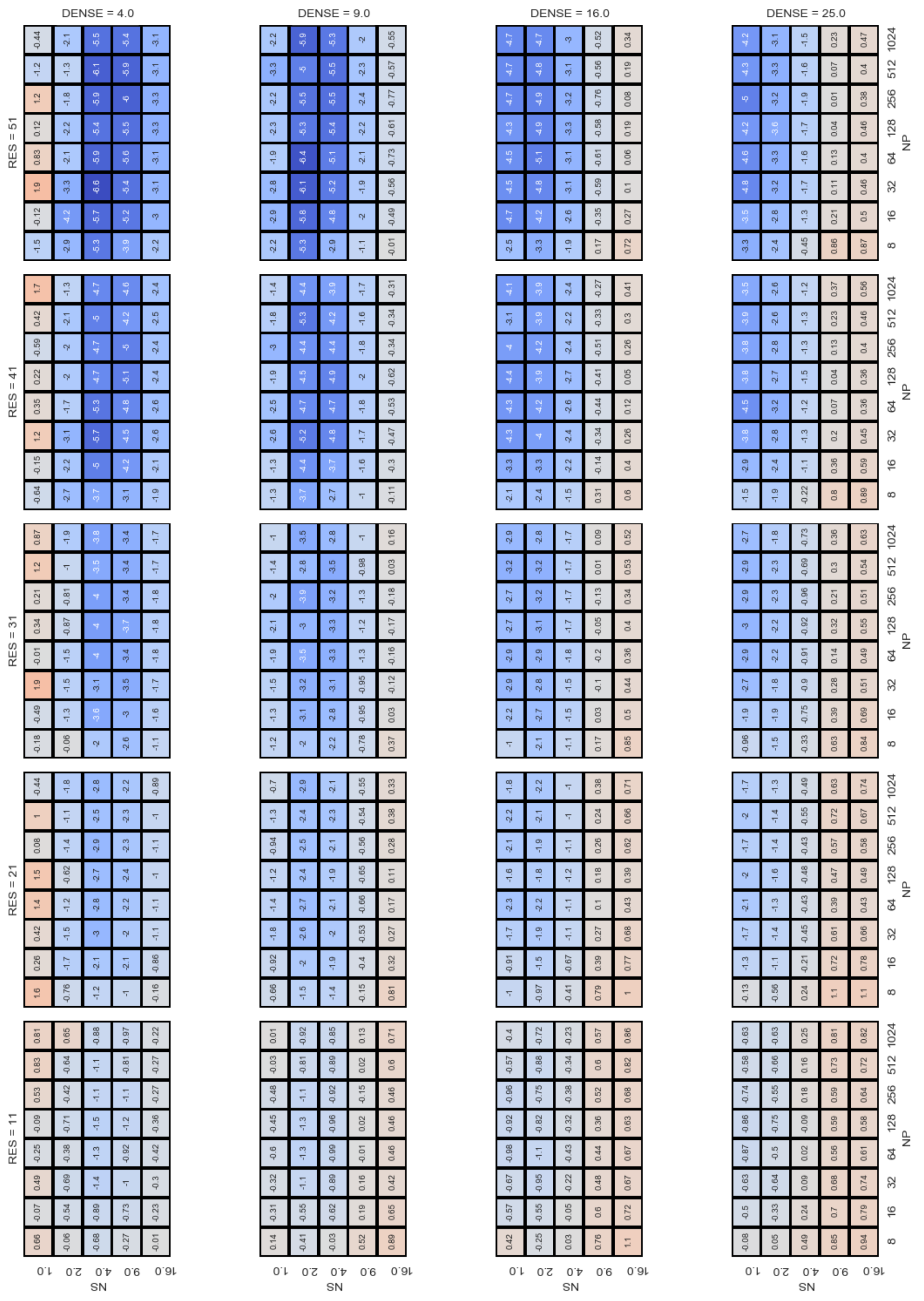


Figure D.10: Heatmap of average iterations required for CSCP convergence over Info-Max for various N_s, resolution $\sqrt{N_g}$, threat parameters N_p, and measurement density M_k for a termination threshold $\varepsilon = 1$.

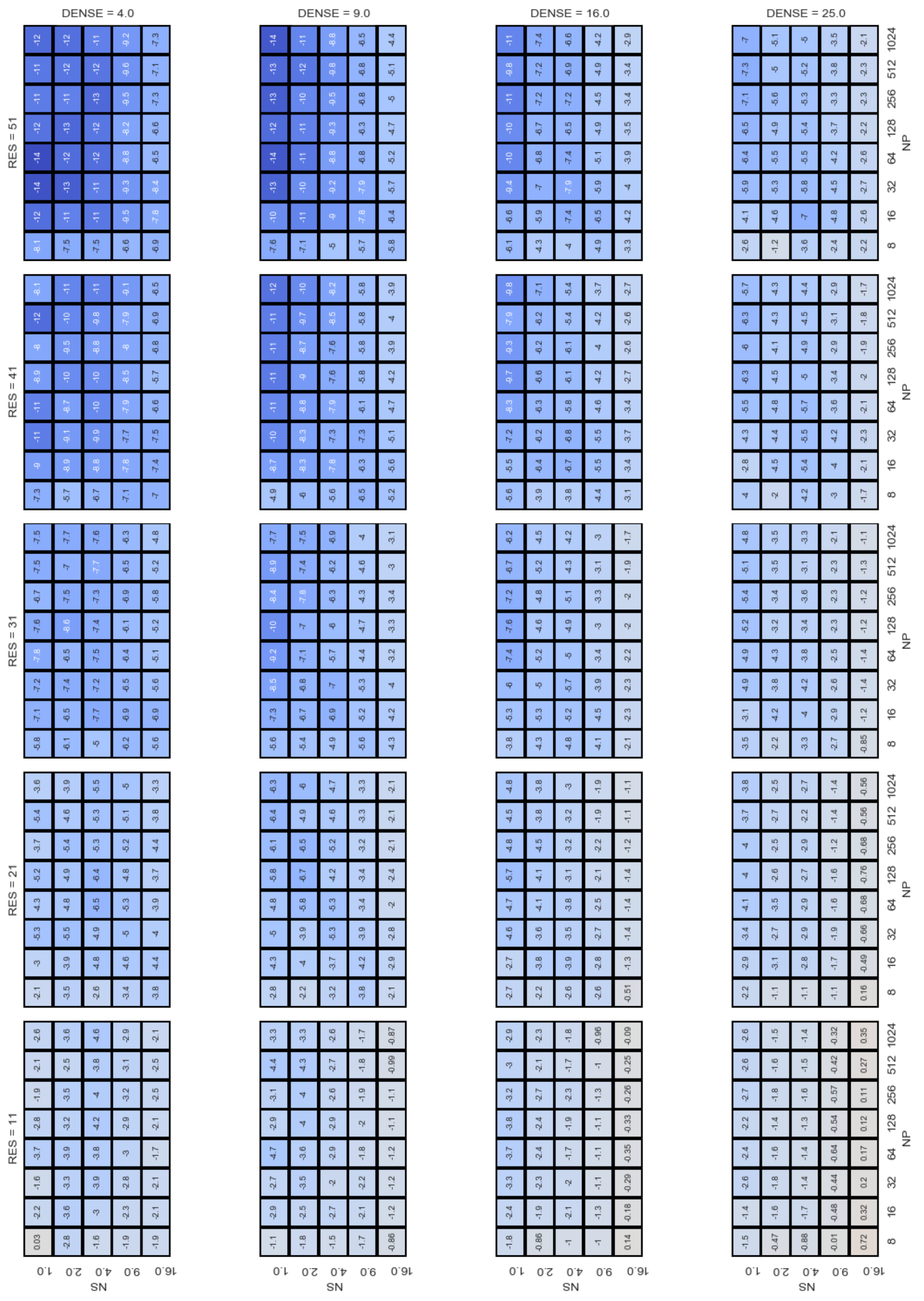


Figure D.11: Heatmap of average iterations required for CSCP convergence over Info-Max for various N_s, resolution $\sqrt{N_g}$, threat parameters N_p, and measurement density M_k for a termination threshold $\varepsilon = 0.1$.

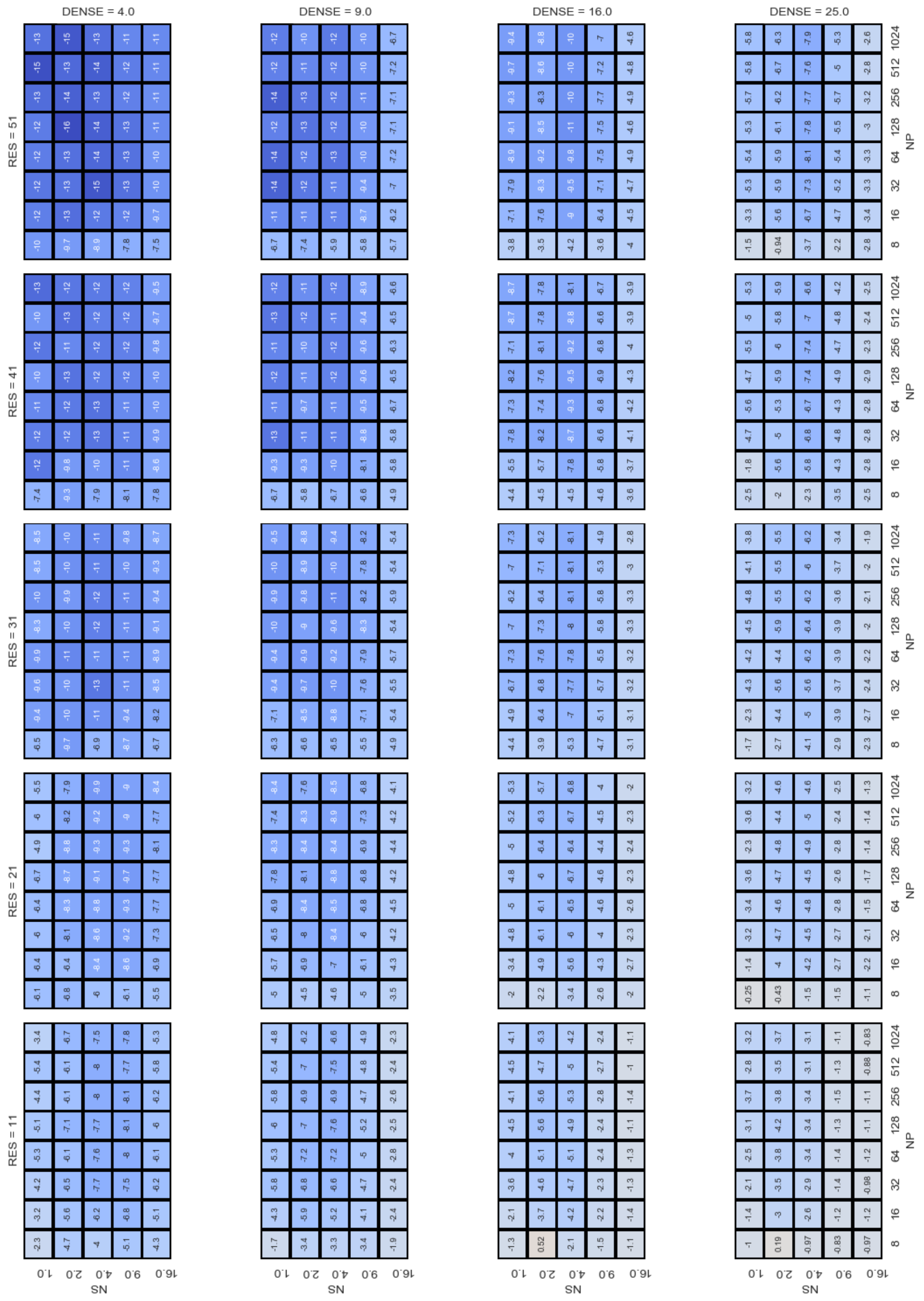


Figure D.12: *Heatmap of average iterations required for CSCP convergence over Info-Max for various N_s, resolution $\sqrt{N_g}$, threat parameters N_p, and measurement density M_k for a termination threshold $\varepsilon = 0.01$.*

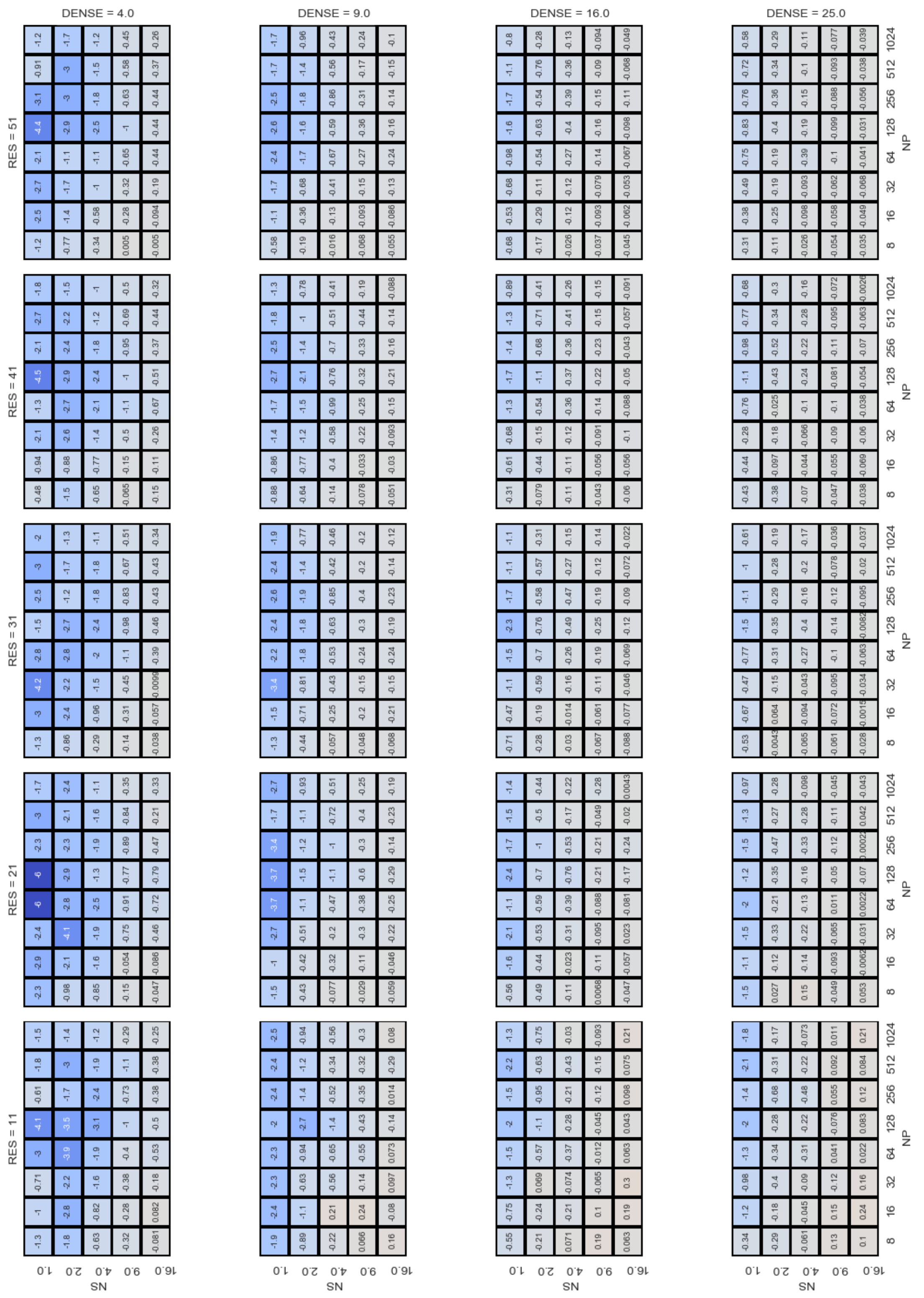


Figure D.13: Heatmap of average error incurred for CSCP over Info-Max for various N_s, resolution $\sqrt{N_g}$, threat parameters N_p, and measurement density M_k for a termination threshold $\varepsilon = 1$.

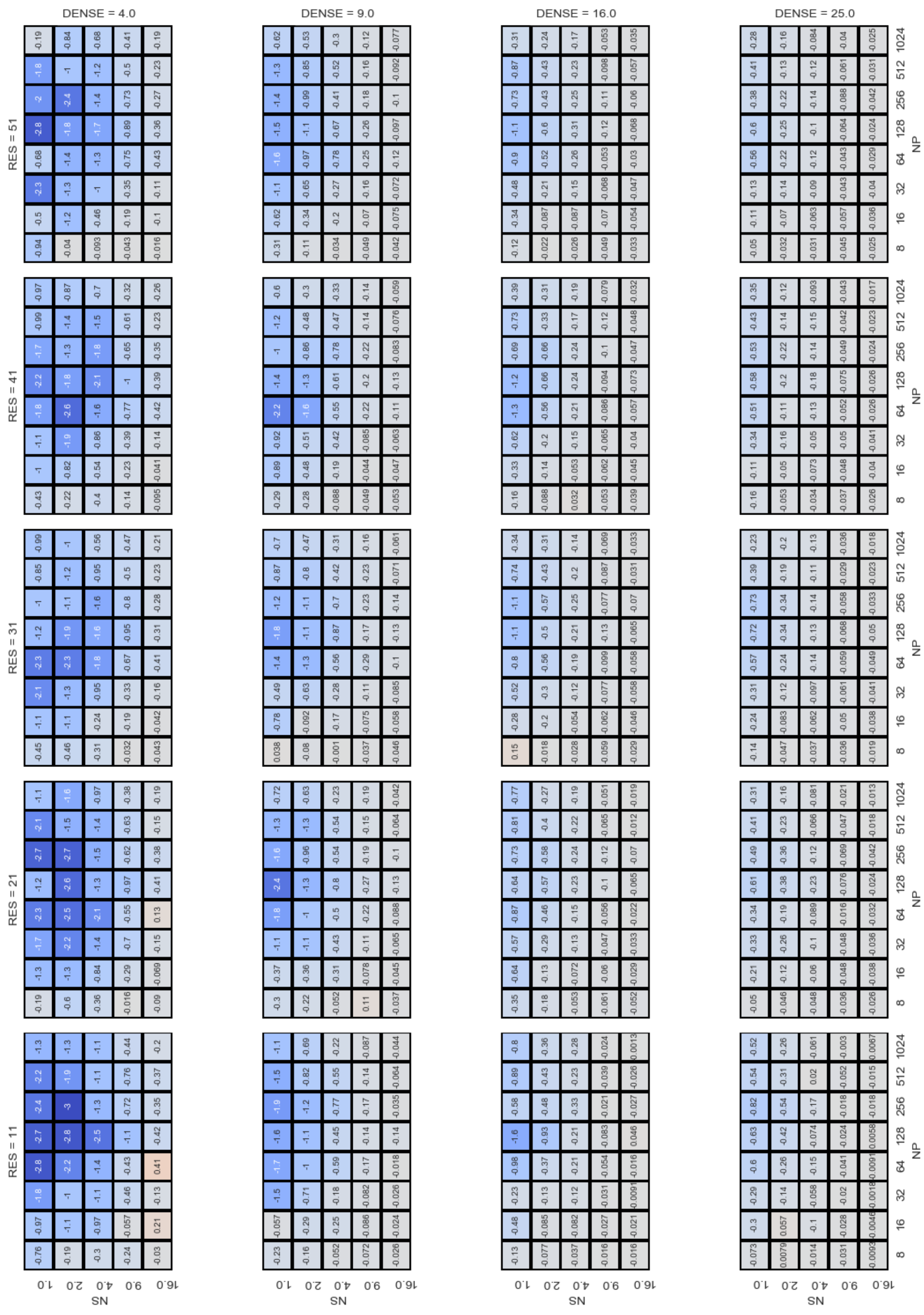


Figure D.14: Heatmap of average error incurred for CSCP over Info-Max for various N_s, resolution $\sqrt{N_g}$, threat parameters N_p, and measurement density M_k for a termination threshold $\varepsilon = 0.1$.

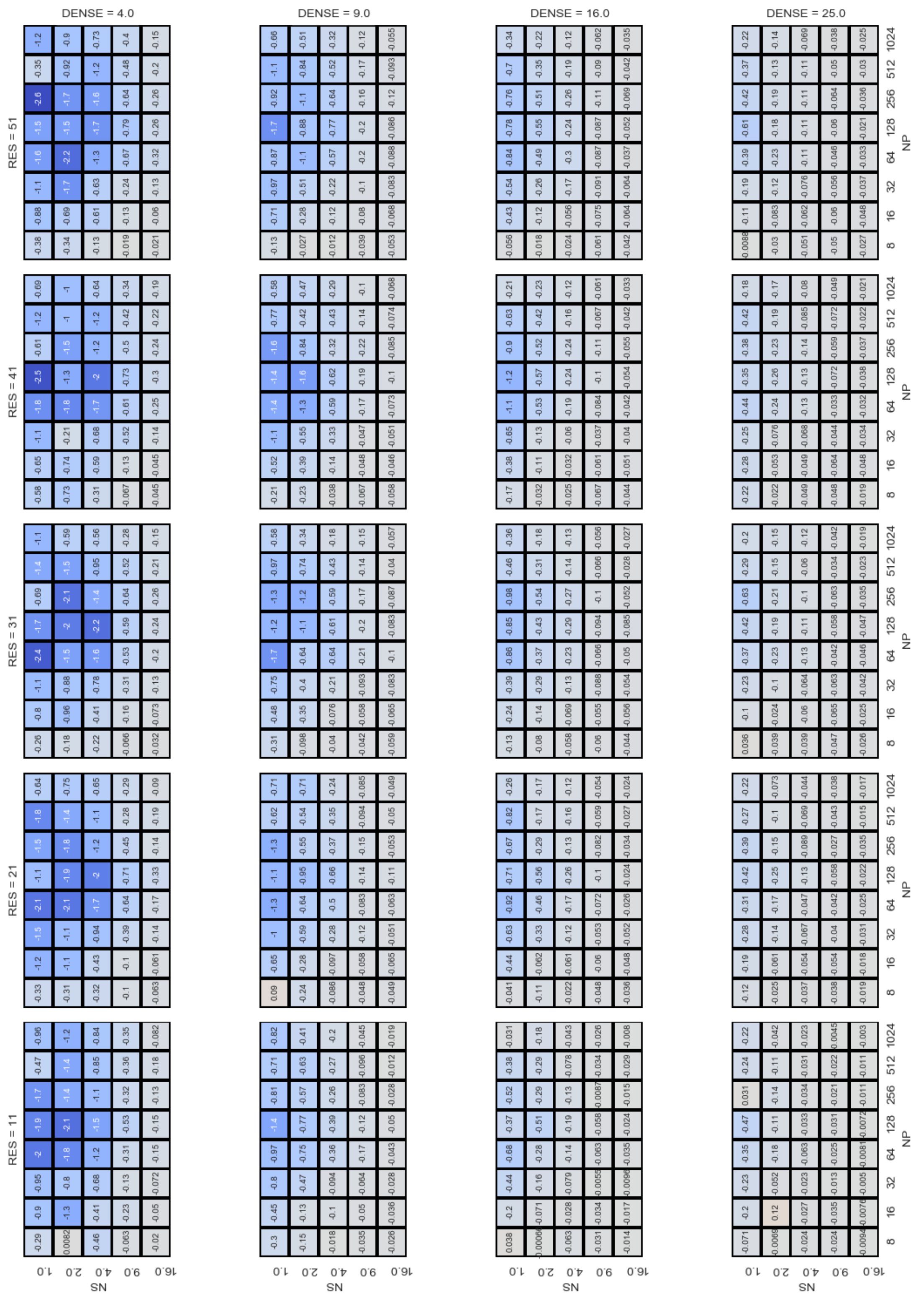


Figure D.15: Heatmap of average error incurred for CSCP over Info-Max for various N_s, resolution $\sqrt{N_g}$, threat parameters N_p, and measurement density M_k for a termination threshold $\varepsilon = 0.01$.

Bibliography

A. B. Abdessalem, N. Dervilis, D. J. Wagg, and K. Worden. Automatic Kernel Selection for Gaussian Processes Regression with Approximate Bayesian Computation and Sequential Monte Carlo. *Frontiers in Built Environment*, 3, 2017. ISSN 2297-3362. doi: 10.3389/fbuil.2017.00052.

S. Aggarwal and N. Kumar. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Computer Communications*, 149:270–299, Jan. 2020. ISSN 01403664. doi: 10.1016/j.comcom.2019.10.014.

W. H. Al-Sabban, L. F. Gonzalez, and R. N. Smith. Wind-energy based path planning for Unmanned Aerial Vehicles using Markov Decision Processes. In *2013 IEEE International Conference on Robotics and Automation*, pages 784–789, May 2013. doi: 10.1109/ICRA.2013.6630662.

T. Allen, A. Hill, J. Underwood, and S. Scheding. Dynamic path planning with multi-agent data fusion - The Parallel Hierarchical Replanner. In *2009 IEEE International Conference on Robotics and Automation*, pages 3245–3250, May 2009. doi: 10.1109/ROBOT.2009.5152883.

M. Alodat and M. K. Shakhatreh. Gaussian process regression with skewed errors. *Journal of Computational and Applied Mathematics*, 370:112665, May 2020. ISSN 03770427. doi: 10.1016/j.cam.2019.112665.

S. Anavatti, S. Francis, and M. Garratt. Path-planning modules for Autonomous Vehicles: Current status and challenges. pages 205–214, Oct. 2015. doi: 10.1109/ICAMIMIA.2015.7508033.

G. Andrew, J. Carlin, H. Stern, D. Dunson, A. Vehtari, and D. Rubin. *Bayesian Data Analysis, Third Edition*. Chapman & Hall/CRC, Boca Raton, FL, 2014. ISBN 978-1-4398-4095-5.

D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA

’07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 9780898716245.

J. Banfi, J. Guzzi, A. Giusti, L. Gambardella, and G. A. Di Caro. Fair multi-target tracking in cooperative multi-robot systems. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5411–5418. IEEE, 2015.

E. Blasch, D. A. Lambert, and E. Bosse. *High-Level Information Fusion Management and Systems Design*. Artech House, 2012. ISBN 978-1-60807-151-7.

E. Blasch, J. J. Salerno, I. Kadar, S. J. Yang, L. Fenstermacher, M. R. Endsley, and L. Grewe. Summary of human social, cultural, behavioral (hscb) modeling for information fusion panel discussion. *Proceedings of SPIE - The International Society for Optical Engineering*, 8745: 87451I, May 2013.

E. Blasch, S. Liu, Z. Liu, and Y. Zheng. Deep Learning Measures of Effectiveness. In *NAECON 2018 - IEEE National Aerospace and Electronics Conference*, pages 254–261, July 2018. doi: 10.1109/NAECON.2018.8556808.

E. Blasch, R. Cruise, A. Aved, U. Majumder, and T. Rovito. Methods of AI for Multimodal Sensing and Action for Complex Situations. *AI Magazine*, 40(4):50–65, 2019.

M. Blum and M. Riedmiller. Optimization of Gaussian Process Hyperparameters using Rprop. *Computational Intelligence*, page 6, 2013.

S. Calinon, F. Guenter, and A. Billard. On Learning, Representing, and Generalizing a Task in a Humanoid Robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, Apr. 2007. ISSN 1941-0492. doi: 10.1109/TSMCB.2006.886952.

R. J. G. B. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-37456-2.

G. C. Cawley and N. L. C. Talbot. Preventing Over-Fitting during Model Selection via Bayesian Regularisation of the Hyper-Parameters. page 21.

S. Chakravorty and R. Saha. Simultaneous planning localization and mapping: A hybrid Bayesian/ frequentist approach. In *2008 American Control Conference*, pages 1226–1231, June 2008. doi: 10.1109/ACC.2008.4586660.

J. Chen, T. Li, T. Shu, and C. W. De Silva. Rapidly-Exploring Tree With Linear Reduction: A Near-Optimal Approach for Spatiotemporal Sensor Deployment in Aquatic Fields Using Minimal Sensor Nodes. *IEEE Sensors Journal*, 18(24):10225–10239, Dec. 2018. ISSN 1558-1748. doi: 10.1109/JSEN.2018.2874393.

H.-T. Chiang, N. Rackley, and L. Tapia. Runtime SES planning: Online motion planning in environments with stochastic dynamics and uncertainty. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4802–4809, Oct. 2016. doi: 10.1109/IROS.2016.7759705.

S. Chung, J. Lim, K. J. Noh, G. Kim, and H. Jeong. Sensor Data Acquisition and Multimodal Sensor Fusion for Human Activity Recognition Using Deep Learning. *Sensors*, 19(7):1716, Jan. 2019. doi: 10.3390/s19071716.

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

B. Cooper and R. V. Cowlagi. Decentralized Interactive Planning and Sensing in an Unknown Spatiotemporal Threat Field. In *2019 Sixth Indian Control Conference (ICC)*, pages 110–115, Dec. 2019a. doi: 10.1109/ICC47138.2019.9123221.

B. Cooper and R. V. Cowlagi. Interactive Planning and Sensing in Spatiotemporally Varying Uncertain Environments. In *2019 18th European Control Conference (ECC)*, pages 2058–2064, June 2019b. doi: 10.23919/ECC.2019.8796184.

B. S. Cooper and R. V. Cowlagi. Interactive planning and sensing in unknown static environments with task-driven sensor placement. *Automatica*, 105:391–398, July 2019c. ISSN 00051098. doi: 10.1016/j.automatica.2019.04.014.

M. M. Costa and M. F. Silva. A Survey on Path Planning Algorithms for Mobile Robots. In *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 1–7, Apr. 2019. doi: 10.1109/ICARSC.2019.8733623.

R. V. Cowlagi. Multiresolution path-planning with traversal costs based on time-varying spatial fields. In *53rd IEEE Conference on Decision and Control*, pages 3745–3750, Dec. 2014. doi: 10.1109/CDC.2014.7039972.

A. C. Damianou and N. D. Lawrence. Deep Gaussian Processes. In *16th International Conference on Artificial Intelligence and Statistics*, volume 31, Scottsdale, AZ, USA, 2013. W&CP.

M. A. Demetriou and D. Uciński. State estimation of spatially distributed processes using mobile sensing agents. In *Proceedings of the 2011 American Control Conference*, pages 1770–1776, June 2011. doi: 10.1109/ACC.2011.5991065.

X. Deng, Y. Jiang, L. Yang, M. Lin, L. Yi, and M. Wang. Data Fusion Based Coverage Optimization in Heterogeneous Sensor Networks: A Survey. *Information Fusion*, 52, Dec. 2018. doi: 10.1016/j.inffus.2018.11.020.

R. Du and R. V. Cowlagi. Interactive sensing and path-planning with incremental 3D path repair for a quadrotor UAV in cluttered and partially known environments. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 933–938, Dec. 2017. doi: 10.1109/CDC.2017.8263778.

J. Dunik, O. Straka, M. Simandl, and E. Blasch. Random-point-based filters: Analysis and comparison in target tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 51(2): 1403–1421, Apr. 2015. ISSN 1557-9603. doi: 10.1109/TAES.2014.130136.

H. Durrant-Whyte and T. C. Henderson. Multisensor data fusion. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, pages 867–896. Springer International Publishing, Cham, 2016. ISBN 978-3-319-32552-1. doi: 10.1007/978-3-319-32552-1\%00835.

D. Duvenaud, J. R. Lloyd, R. Grosse, J. B. Tenenbaum, and Z. Ghahramani. Structure Discovery in Nonparametric Regression through Compositional Kernel Search. page 9.

M. Elbanhawi and M. Simic. Sampling-Based Robot Motion Planning: A Review. *IEEE Access*, 2:56–77, 2014. ISSN 2169-3536. doi: 10.1109/ACCESS.2014.2302442.

M. Farsi, M. A. Elhosseini, M. Badawy, H. Arafat Ali, and H. Zain Eldin. Deployment Techniques in Wireless Sensor Networks, Coverage and Connectivity: A Survey. *IEEE Access*, 7:28940–28954, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2902072.

M. Fernández-Delgado, M. Sirsat, E. Cernadas, S. Alawadi, S. Barro, and M. Febrero-Bande. An extensive experimental survey of regression methods. *Neural Networks*, 111:11–34, Mar. 2019. ISSN 08936080. doi: 10.1016/j.neunet.2018.12.010.

S. Fine. Efficient SVM Training Using Low-Rank Kernel Representations. page 22.

D. Garagić, J. Peskoe, F. Liu, M. S. Claffey, P. Bendich, J. Hineman, N. Borggren, J. Harer, P. Zulch, and B. J. Rhodes. Upstream fusion of multiple sensing modalities using machine learning and topological analysis: An initial exploration. In *2018 IEEE Aerospace Conference*, pages 1–8, Mar. 2018. doi: 10.1109/AERO.2018.8396737.

S. S. M. Gharib and P. Esmaili. ANFIS Based UKF-SLAM Path Planning Method. In *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pages 1–5, Oct. 2019. doi: 10.1109/ISMSIT.2019.8932958.

K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, Oct. 2017. ISSN 2162-237X, 2162-2388. doi: 10.1109/TNNLS.2016.2582924.

T. Gu, K. Song, and C. Miao. A Motion Planning Algorithm Based on Uncertainty Prediction. In *2013 International Conference on Information Science and Cloud Computing*, pages 1–6, Dec. 2013. doi: 10.1109/ISCC.2013.8.

M. Guo and M. M. Zavlanos. Probabilistic Motion Planning Under Temporal Tasks and Soft Constraints. *IEEE Transactions on Automatic Control*, 63(12):4051–4066, Dec. 2018. ISSN 1558-2523. doi: 10.1109/TAC.2018.2799561.

W. Guo, J. Wang, and S. Wang. Deep Multimodal Representation Learning: A Survey. *IEEE Access*, 7:63373–63394, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2916887.

E. Hasanbelliu, K. Kampa, J. C. Principe, and J. T. Cobb. Online learning using a bayesian surprise metric. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2012. doi: 10.1109/IJCNN.2012.6252734.

R. Herbrich, N. D. Lawrence, and M. Seeger. Fast Sparse Gaussian Process Methods: The Informative Vector Machine. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 625–632. MIT Press, 2003.

Y. Huang and K. Gupta. RRT-SLAM for motion planning with motion and map uncertainty for robot exploration. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1077–1082, Sept. 2008. doi: 10.1109/IROS.2008.4651183.

H. Igarashi. Motion planning of a mobile robot as a discrete optimization problem. In *Proceedings of the 2001 IEEE International Symposium on Assembly and Task Planning (ISATP2001). Assembly and Disassembly in the Twenty-First Century. (Cat. No.01TH8560)*, pages 1–6, May 2001. doi: 10.1109/ISATP.2001.928957.

A. S. H. H. V. Injarapu and S. K. Gawre. A survey of autonomous mobile robot path planning approaches. In *2017 International Conference on Recent Innovations in Signal Processing and Embedded Systems (RISE)*, pages 624–628, Oct. 2017. doi: 10.1109/RISE.2017.8378228.

S. A. Israel and E. Blasch. Context Assumptions for Threat Assessment Systems. In L. Snidaro, J. García, J. Llinas, and E. Blasch, editors, *Context-Enhanced Information Fusion*, pages 99–124. Springer International Publishing, Cham, 2016. ISBN 978-3-319-28969-4 978-3-319-28971-7. doi: 10.1007/978-3-319-28971-7_5.

X. Ji and J. Li. Online Motion Planning for UAV under Uncertain Environment. In *2015 8th International Symposium on Computational Intelligence and Design (ISCID)*, volume 2, pages 514–517, Dec. 2015. doi: 10.1109/ISCID.2015.178.

B. Kahler and E. Blasch. Predicted radar/optical feature fusion gains for target identification. In

Proceedings of the IEEE 2010 National Aerospace Electronics Conference, pages 405–412, July 2010. doi: 10.1109/NAECON.2010.5712986.

M. Kamezaki, J. Yang, H. Iwata, and S. Sugano. Visibility enhancement using autonomous multicamera controls with situational role assignment for teleoperated work machines. *Journal of Field Robotics*, 33(6):802–824, 2016.

L. Kaufman and P. J. Rousseeuw. Partitioning around medoids (program pam). *Finding groups in data: an introduction to cluster analysis*, 344:68–125, 1990.

H. Kawano. Method for applying reinforcement learning to motion planning and control of under-actuated underwater vehicle in unknown non-uniform sea flow. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 996–1002, Aug. 2005. doi: 10.1109/IROS.2005.1544973.

A. Khan, B. Rinner, and A. Cavallaro. Cooperative robots to observe moving targets. *IEEE transactions on cybernetics*, 48(1):187–198, 2016.

J. Ko, D. J. Klein, D. Fox, and D. Haehnel. GP-UKF: Unscented kalman filters with Gaussian process prediction and observation models. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1901–1907, San Diego, CA, Oct. 2007. IEEE. ISBN 978-1-4244-0911-2. doi: 10.1109/IROS.2007.4399284.

A. Krause, A. Singh, and C. Guestrin. Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. page 50.

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 (2), 2008.

A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. *Robust Sensor Placements at Informative and Communication-Efficient Locations*. 2010.

P. Krishnamurthy and F. Khorrami. Optimal Sensor Placement for Monitoring of Spatial Networks. *IEEE Transactions on Automation Science and Engineering*, 15(1):33–44, Jan. 2018. ISSN 1558-3783. doi: 10.1109/TASE.2016.2573818.

H. Kurniawati, T. Bandyopadhyay, and N. M. Patrikalakis. Global Motion Planning under Uncertain Motion, Sensing, and Environment Map. page 8.

M. W. Y. Lam, X. Chen, S. Hu, J. Yu, X. Liu, and H. Meng. Gaussian Process Lstm Recurrent Neural Network Language Models for Speech Recognition. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7235–7239, May 2019. doi: 10.1109/ICASSP.2019.8683660.

S. M. LaValle. Motion Planning. *IEEE Robotics Automation Magazine*, 18(1):79–89, Mar. 2011. ISSN 1558-223X. doi: 10.1109/MRA.2011.940276.

J. Li, B. Dai, D. Liu, Z. Zhao, and J. Ren. An optimal sampling-based path planning under uncertainty based on linear quadratic regulator. In *2017 2nd International Conference on Robotics and Automation Engineering (ICRAE)*, pages 84–88, Dec. 2017. doi: 10.1109/ICRAE.2017.8291358.

J. Li, S. X. Yang, and Z. Xu. A Survey on Robot Path Planning using Bio-inspired Algorithms. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2111–2116, Dec. 2019. doi: 10.1109/ROBIO49542.2019.8961498.

S. Li, H. Zhang, S. Liu, and Z. Zhang. Optimal sensor placement using FRFs-based clustering method. *Journal of sound and vibration*, 385:69–80, 2016. ISSN 0022-460X.

Z. Li, X. Hong, K. Hao, L. Chen, and B. Huang. Gaussian process regression with heteroscedastic noises — A machine-learning predictive variance approach. *Chemical Engineering Research and Design*, 157:162–173, May 2020. ISSN 02638762. doi: 10.1016/j.cherd.2020.02.033.

K. Liu, Y. Zhang, A. Dobson, and D. Berenson. Asymptotically Near-Optimal Methods for Kinodynamic Planning With Initial State Uncertainty. *IEEE Robotics and Automation Letters*, 4(2): 2124–2131, Apr. 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2899931.

J. Llinas, L. Snidaro, J. García, and E. Blasch. Context and Fusion: Definitions, Terminology. In L. Snidaro, J. García, J. Llinas, and E. Blasch, editors, *Context-Enhanced Information Fusion*, pages 3–23. Springer International Publishing, Cham, 2016. ISBN 978-3-319-28969-4 978-3-319-28971-7. doi: 10.1007/978-3-319-28971-7_1.

C.-K. Lu and P. Shafto. Multi-source Deep Gaussian Process Kernel Learning. *arXiv:2002.02826 [cond-mat, stat]*, Feb. 2020.

T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser. Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems*, 86:13–28, Dec. 2016. ISSN 09218890. doi: 10.1016/j.robot.2016.08.001.

R. Madankan, S. Pouget, P. Singla, M. Bursik, J. Dehn, M. Jones, A. Patra, M. Pavolonis, E. 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, Aug. 2014. ISSN 00219991. doi: 10.1016/j.jcp.2013.11.032.

S. Martínez and F. Bullo. Optimal sensor placement and motion coordination for target tracking. *Automatica*, 42(4):661–668, Apr. 2006. ISSN 00051098. doi: 10.1016/j.automatica.2005.12.018.

A. Mavrinac and X. Chen. Modeling coverage in camera networks: A survey. *International journal of computer vision*, 101(1):205–226, 2013.

D. P. Mesquita, J. P. Gomes, F. Corona, A. H. Souza, and J. S. Nobre. Gaussian kernels for incomplete data. *Applied Soft Computing*, 77:356–365, Apr. 2019. ISSN 15684946. doi: 10.1016/j.asoc.2019.01.022.

P. Mohajerin Esfahani, D. Chatterjee, and J. Lygeros. Motion Planning for Continuous-Time Stochastic Processes: A Dynamic Programming Approach. *IEEE Transactions on Automatic Control*, 61(8):2155–2170, Aug. 2016. ISSN 1558-2523. doi: 10.1109/TAC.2015.2500638.

M. Mohanan and A. Salgoankar. A survey of robotic motion planning in dynamic environments. *Robotics and Autonomous Systems*, 100:171–185, Feb. 2018. ISSN 09218890. doi: 10.1016/j.robot.2017.10.011.

J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal Deep Learning. page 8.

L. V. Nguyen, S. Kodagoda, R. Ranasinghe, and G. Dissanayake. Adaptive Placement for Mobile Sensors in Spatial Prediction Under Locational Errors. *IEEE Sensors Journal*, 17(3):794–802, Feb. 2017. ISSN 1558-1748. doi: 10.1109/JSEN.2016.2633958.

B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, Mar. 2016. ISSN 2379-8904. doi: 10.1109/TIV.2016.2578706.

B. Patle, G. Babu L, A. Pandey, D. Parhi, and A. Jagadeesh. A review: On path planning strategies for navigation of mobile robot. *Defence Technology*, 15(4):582–606, Aug. 2019. ISSN 22149147. doi: 10.1016/j.dt.2019.04.011.

D. Praveen Kumar, T. Amgoth, and C. S. R. Annavarapu. Machine learning algorithms for wireless sensor networks: A survey. *Information Fusion*, 49:1–25, Sept. 2019. ISSN 15662535. doi: 10.1016/j.inffus.2018.09.013.

L. Quan, L. Han, B. Zhou, S. Shen, and F. Gao. Survey of UAV motion planning. *IET Cyber-systems and Robotics*, 2(1):14–21, 2020. ISSN 2631-6315. doi: 10.1049/iet-csr.2020.0004.

G. R. and V. Uma. Ontology based knowledge representation technique, domain modeling languages and planners for robotic path planning: A survey. *ICT Express*, 4(2):69–74, June 2018. ISSN 24059595. doi: 10.1016/j.icte.2018.04.008.

D. Ramsden. OPTIMIZATION APPROACHES TO SENSOR PLACEMENT PROBLEMS. page 80.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass, 2006. ISBN 978-0-262-18253-9.

S. Reece and S. Roberts. An introduction to Gaussian processes for the Kalman filter expert. In *2010 13th International Conference on Information Fusion*, pages 1–9, Edinburgh, July 2010. IEEE. ISBN 978-0-9824438-1-1. doi: 10.1109/ICIF.2010.5711863.

S. Rhode. Non-stationary Gaussian process regression applied in validation of vehicle dynamics models. *Engineering Applications of Artificial Intelligence*, 93:103716, Aug. 2020. ISSN 09521976. doi: 10.1016/j.engappai.2020.103716.

S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain. Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110550, Feb. 2013. ISSN 1364-503X, 1471-2962. doi: 10.1098/rsta.2011.0550.

C. Robin and S. Lacroix. Multi-robot target detection and tracking: taxonomy and survey. *Autonomous Robots*, 40(4):729–760, 2016.

R. Sarrate, J. Blesa, and F. Nejjari. Clustering techniques applied to sensor placement for leak detection and location in water distribution networks. In *22nd Mediterranean Conference on Control and Automation*, pages 109–114, Palermo, Italy, June 2014. IEEE. ISBN 978-1-4799-5901-3 978-1-4799-5900-6. doi: 10.1109/MED.2014.6961356.

M. Schürch, D. Azzimonti, A. Benavoli, and M. Zaffalon. Recursive estimation for sparse Gaussian process regression. *Automatica*, 120:109127, Oct. 2020. ISSN 00051098. doi: 10.1016/j.automatica.2020.109127.

O. Sharma, N. C. Sahoo, and N. B. Puhan. A Survey on Smooth Path Generation Techniques for Nonholonomic Autonomous Vehicle Systems. In *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 5167–5172, Oct. 2019. doi: 10.1109/IECON.2019.8926946.

D. Shen, E. Blasch, P. Zulch, M. Distasio, R. Niu, J. Lu, Z. Wang, and G. Chen. A joint manifold leaning-based framework for heterogeneous upstream data fusion. *Journal of Algorithms & Computational Technology*, 12(4):311–332, Dec. 2018. ISSN 1748-3026. doi: 10.1177/1748301818791507.

V. N. Sichkar. Reinforcement Learning Algorithms in Global Path Planning for Mobile Robot. In *2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, pages 1–5, Mar. 2019. doi: 10.1109/ICIEAM.2019.8742915.

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. In *2006*

IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2436–2442, Oct. 2006. doi: 10.1109/IROS.2006.281685.

L. Snidaro, J. García, J. Llinas, and E. Blasch, editors. *Context-Enhanced Information Fusion: Boosting Real-World Performance with Domain Knowledge*. Advances in Computer Vision and Pattern Recognition. Springer International Publishing, Cham, 2016. ISBN 978-3-319-28969-4 978-3-319-28971-7. doi: 10.1007/978-3-319-28971-7.

B. Song, G. Qi, and L. Xu. A Survey of Three-Dimensional Flight Path Planning for Unmanned Aerial Vehicle. In *2019 Chinese Control And Decision Conference (CCDC)*, pages 5010–5015, June 2019. doi: 10.1109/CCDC.2019.8832890.

H. Soni, V. Gupta, and R. Kumar. Motion Planning using Reinforcement Learning for Electric Vehicle Battery optimization(EVBO). In *2019 International Conference on Power Electronics, Control and Automation (ICPECA)*, pages 1–6, Nov. 2019. doi: 10.1109/ICPECA47973.2019.8975684.

S. Stanton, W. Maddox, I. Delbridge, and A. G. Wilson. Kernel interpolation for scalable online gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 3133–3141. PMLR, 2021.

F. Stulp and O. Sigaud. Many regression algorithms, one unified model: A review. *Neural Networks*, 69:60–79, Sept. 2015. ISSN 08936080. doi: 10.1016/j.neunet.2015.05.005.

C. Sun, Y. Yu, V. O. Li, and J. C. Lam. Optimal Multi-type Sensor Placements in Gaussian Spatial Fields for Environmental Monitoring. In *2018 IEEE International Smart Cities Conference (ISC2)*, pages 1–8, Sept. 2018. doi: 10.1109/ISC2.2018.8656676.

H. Sun, W. Deng, S. Zhang, S. Wang, and Y. Zhang. Trajectory planning for vehicle autonomous driving with uncertainties. In *Proceedings 2014 International Conference on Informative and Cybernetics for Computational Social Systems (ICCSS)*, pages 34–38, Oct. 2014. doi: 10.1109/ICCSS.2014.6961812.

Y. Sung, A. K. Budhiraja, R. K. Williams, and P. Tokekar. Distributed simultaneous action and target assignment for multi-robot multi-target tracking. In *2018 IEEE International conference on robotics and automation (ICRA)*, pages 3724–3729. IEEE, 2018.

T. Tao, Y. Huang, F. Sun, and T. Wang. Motion Planning for SLAM Based on Frontier Exploration. In *2007 International Conference on Mechatronics and Automation*, pages 2120–2125, Aug. 2007. doi: 10.1109/ICMA.2007.4303879.

E. L. Thorn. Autonomous Motion Planning Using a Predictive Temporal Method. page 171.

S. Thrun. Learning Occupancy Grid Maps With Forward Sensor Models. page 28.

S. Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3), Mar. 2002. ISSN 00010782. doi: 10.1145/504729.504754.

M. Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pages 567–574. PMLR, 2009.

A. Toriz Palacios and A. Sánchez López. A New Strategy for Exploring Unknown Environments for the SLAM Problem. In *2017 Sixteenth Mexican International Conference on Artificial Intelligence (MICAI)*, pages 9–14, Oct. 2017. doi: 10.1109/MICAI-2017.2017.00010.

V. Tresp. A Bayesian Committee Machine. *Neural Computation*, 12(11):2719–2741, Nov. 2000. ISSN 0899-7667, 1530-888X. doi: 10.1162/089976600300014908.

D. Uciński. Sensor network scheduling for identification of spatially distributed processes. *International Journal of Applied Mathematics and Computer Science*, 22(1):25–40, Mar. 2012. ISSN 1641-876X. doi: 10.2478/v10006-012-0002-0.

L. Van Nguyen, S. Kodagoda, R. Ranasinghe, and G. Dissanayake. Locational optimization based sensor placement for monitoring Gaussian processes modeled spatial phenomena. In *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*, pages 1706–1711, June 2013. doi: 10.1109/ICIEA.2013.6566643.

S. Vasudevan. Data fusion with Gaussian processes. *Robotics and Autonomous Systems*, 60(12): 1528–1544, Dec. 2012. ISSN 09218890. doi: 10.1016/j.robot.2012.08.006.

J. K. Verma and V. Ranga. Target tracking with cooperative networked robots. In *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 981–985, 2020. doi: 10.1109/SPIN48934.2020.9071411.

H. Wang, Y.-M. Zhang, J.-X. Mao, and H.-P. Wan. A probabilistic approach for short-term prediction of wind gust speed using ensemble learning. *Journal of Wind Engineering and Industrial Aerodynamics*, 202:104198, July 2020. ISSN 0167-6105. doi: 10.1016/j.jweia.2020.104198.

J. Wang and X. Su. An improved K-Means clustering algorithm. In *2011 IEEE 3rd International Conference on Communication Software and Networks*, pages 44–46, May 2011. doi: 10.1109/ICCSN.2011.6014384.

C. K. I. Williams and M. Seeger. Using the Nyström Method to Speed Up Kernel Machines. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep Kernel Learning. *arXiv:1511.02222 [cs, stat]*, Nov. 2015.

J. Xie, X. Shen, Z. Wang, and Y. Zhu. Gaussian Process Fusion for Multisensor Nonlinear Dynamic Systems. In *2018 37th Chinese Control Conference (CCC)*, pages 4124–4129, July 2018. doi: 10.23919/ChiCC.2018.8483873.

R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005. doi: 10.1109/TNN.2005.845141.

T. Yan, Y. Zhang, and B. Wang. Path Planning for Mobile Robot's Continuous Action Space Based on Deep Reinforcement Learning. In *2018 International Conference on Big Data and Artificial Intelligence (BDAI)*, pages 42–46, June 2018. doi: 10.1109/BDAI.2018.8546675.

Y. Yang, J. Pan, and W. Wan. Survey of optimal motion planning. *IET Cyber-systems and Robotics*, 1(1):13–19, 2019. ISSN 2631-6315. doi: 10.1049/iet-csr.2018.0003.

D. Yoganathan, S. Kondepudi, B. Kalluri, and S. Manthapuri. Optimal sensor placement strategy for office buildings using clustering algorithms. *Energy and buildings*, 158:1206–1225, 2018. ISSN 0378-7788.

M. Younis and K. Akkaya. Strategies and techniques for node placement in wireless sensor networks: A survey. *Ad Hoc Networks*, 6(4):621–655, June 2008. ISSN 15708705. doi: 10.1016/j.adhoc.2007.05.003.

Y. Zhao, Z. Zheng, and Y. Liu. Survey on computational-intelligence-based UAV path planning. *Knowledge-Based Systems*, 158:54–64, Oct. 2018. ISSN 09507051. doi: 10.1016/j.knosys.2018.05.033.

X. Zhou, Z. Yi, Y. Liu, K. Huang, and H. Huang. Survey on path and view planning for UAVs. *Virtual Reality & Intelligent Hardware*, 2(1):56–69, Feb. 2020. ISSN 20965796. doi: 10.1016/j.vrih.2019.12.004.