Path-Based Sensors: Paths as Sensors, Bayesian Updates, and Shannon Information Gathering

Michael Otte, *Member, IEEE,* and Donald Sofge, *Senior Member, IEEE*

***Abstract*—Consider a sensor that reports whether or not an event has occurred *somewhere* along a path, but that has no conception of *where* along the path that event has occurred. We name this type of sensor a *path-based sensor*, and describe the recursive Bayesian update that can be used to calculate posterior beliefs about the presence of a sensor triggering phenomenon given a path-based sensor observation. We show how the Bayesian update can be leveraged to calculate the expected Shannon information that will be gained along a particular path. We formalize two iterative information gathering problems that results from this scenario, and present path-planning algorithms to solve them. These include: (1) gathering information about the path-based sensor triggering phenomena, and (2) assuming the path-based sensor triggering event is "robot destruction," simultaneously gather information about (i) hazards using a path-based sensor and (ii) information about another environmental phenomenon using a standard sensor, such as the locations of search and rescue targets with a camera. We evaluate our methods using Monte Carlo simulations, and observe that they outperform other techniques with respect to the new problems we consider.**

Note to Practitioners: ***Abstract*—This work is motivated by the problem of searching for robot destroying hazards that are otherwise invisible to the robots. That is, we can observe whether or not a robot survives a path, but if a robot is destroyed then we have no idea where along the path its destruction has occurred. A mathematically equivalent problem happens in any scenario in which an agent is equipped with an event sensor that can only be set/triggered once, but that requires post processing to figure out if the sensor has been triggered or not. For example, post processing is needed if the determination of whether or not a biological specimen was obtained requires a manual laboratory inspection. We also consider an extension of the hazard detection problem in which we simultaneously collect information about search-and-rescue victims using a "victim sensor" like a camera. In this problem hazards indirectly affect information gathered about victims because new information about victims is lost whenever a robot is destroyed. We provide algorithms to solve these types of problems, and that work even in cases with noise such that false positives and false negatives are possible. This work is useful in any application where observations take the form of a cumulative "yes" or "no" along a path.**

Primary and Secondary Keywords ***Index Terms*—Primary Topics: Path Planning, Unmanned Autonomous Vehicles, Robots, Mutual information, Information Entropy, Recursive Estimation. Secondary Topic Keywords: Information Gathering, Paths as Sensors, Target Search, Hazard Detection.**

I. Introduction

The use of autonomous robots for scientific, commercial, and government tasks has become increasingly common in the last few years. This is due to increasing availability, better reliability, and lower costs of autonomous platforms and a variety of useful onboard sensors. A broad category of autonomous robot missions involves gathering information about the world. Oceanographers take temperature and salinity readings, biologists retrieve plant and insect specimens, public utility agencies inspect infrastructure, and governments patrol geopolitical boundaries. The use of an autonomous agent for these tasks requires the agent to move to a position, or set of positions, to make observations and/or collect samples.

Previous missions have tended to fall into one of two categories. Category one: the use of onboard sensors like cameras, thermometers, and chemical sensors enables the agent to gather hundreds, thousands, or millions of readings during a mission. Category two: physical sample/specimen collection is performed, and the physical nature of samples, such as size, weight, sampling procedure, restricts the number of samples that can be retrieved per trip. In both of these cases it is possible to estimate when and where each piece of data is collected during a robot's mission.

In this paper we investigate a very different idea. We are interested in the scenario where we *cannot* know when/where along a path a critical piece of data is collected, but we do know the path along which the agent traveled. This scenario occurs when a robot is equipped with an event sensor that can only be set/triggered once, but that requires post processing to discern whether or not that sensor has made been triggered.

Given (A) an event sensor that can only be triggered once and (B) the path along which the sensor has traveled, we can make an observation (C) "the sensor was triggered along the path (or it was not)". Thus, (A) and (B) combine to form another type of sensor that is shaped like the path and provides the observation (C). We call this type of sensor a "*path-based sensor.*"

In this paper we assume that the path-based sensor produces a binary sensor observation—did an event happen along the path? "yes" or "no". We assume a noisy path-based sensor such that both false positives and false negatives may occur.

As with any sensor, path-based sensor readings provide data that can be use to refine our belief about the location of the

M. Otte is with the Department of Aerospace Engineering, University of Maryland, College Park, MD 20742, USA .

D. Sofge is with the U.S. Naval Research Laboratory, Washington, DC 20375, USA.

Manuscript received ??; revised ??

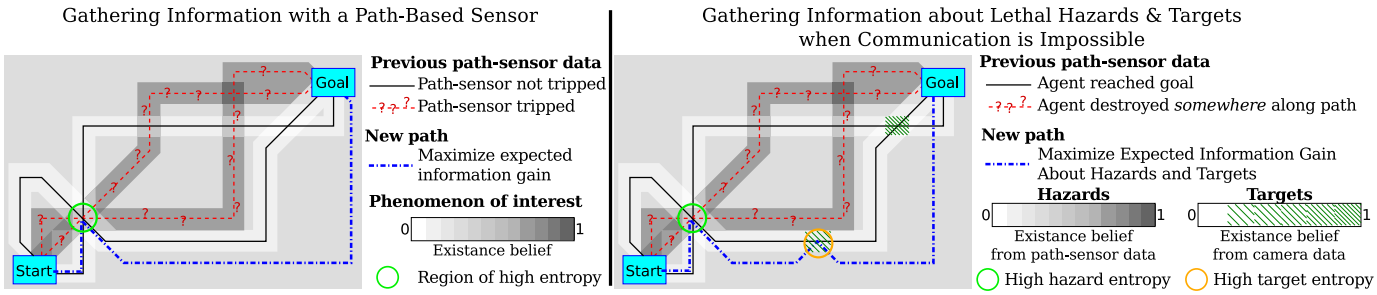

Fig. 1: A path-based sensor returns true or false depending on if an event occurs somewhere along the path or not, respectively. This figure shows two iterative information gathering problems that leverage the path-based sensor idea. Left: a sequence of paths is used to refine beliefs about whether or not locations contain event causing (sensor triggering) phenomena. Right: Search for hazards and targets in an environment without communication. The path-based sensor is used to refine the location of hazards, based on robots successfully returning from a path or disappearing along it, during a mission in which a typical camera sensor is used to simultaneously sense targets.

triggering phenomena. Although a path of any shape can be used for a path-based sensor, we can also design the path to maximize the expected amount of information that will be gained about the locations of the triggering phenomena.

To aid our discussion, we now outline three examples of path-based sensors that will be used throughout this paper.

Ex 1. We desire to find locations at which insects are hatching. An insect sampling robot carries a passive collection device such as a butterfly net through the environment. After the robot returns from each pass through the field, a biologist inspects the net to see if a particular species of insect has been obtained. Paths from which the robot returns with a positive sample are more likely to have visited map regions containing a nest.

Ex 2. Invisible hazards exist in an environment *in which wireless communication is prohibited*. Communication of direct *positive* hazard observations are impossible—the only way a robot can positively "observe" a hazard is to be destroyed by it. *Indirect* information about hazards can be inferred by remembering which path an agent plans to take, and then observing whether or not the agent survives a journey along that path. Agents are less likely to return from paths containing hazards than paths that are hazard free. Note that in this case the path-based sensor observation is "the agent survived the path (or did not)". See Figure 1-Left.

Ex 1 and Ex 2 have identical mathematical formulations despite the fact that Ex 1 involves non-lethal data collection and Ex 2 involves the destruction of the agent itself. This variety helps to showcase the generality of the path-based sensor idea. In either case, an important assumption is that the sensor is active along the entire path. This means that we assume the net cannot be selectively deployed, and that debris from destroyed robots cannot be observed later, e.g., obscured by the environment or removed by an adversary.

A different class of problems involves simultaneously: (i) determining the locations of lethal hazards using a path based sensor while also (ii) using a more typical sensor to gather information about a separate non-lethal environmental phenomenon. Ex 3 illustrates such a scenario.

Ex 3. Autonomous agents are used to help search for human survivors ("targets") in a hazardous environment in which *wireless communication is prohibited*. As agents gather information about survivors' whereabouts, the agent must visit special "uplink sites," like bases or friendly naval vessels, to upload any information that is collected. Target information gathered by an agent is *lost* if that agent is destroyed before reaching an uplink site.

In Ex 3 the objective is to gather a weighted sum of information about both hazards and targets (survivors). See Figure 1-Right. Any calculation of the expected information that will be gathered along a path involves an integral over two mutually exclusive cases: either (1) the robot survives or (2) the robot does not survive. Hazards affect the information gathering task in two completely different ways. First, as in Ex 2, either the destruction (or not) of the agent itself can provide information about the *hazard* locations. Second, the probability the robot is destroyed along a particular path affects the amount of information we expect to gather about *targets* along that path.

Our work is grounded in Bayesian probability and the ideas of Shannon information theory. Given the ability to plan a robot's path, it is possible to maximize the *mutual information* [1], [2] between path-based sensor readings and our existing beliefs. In practice, this means that we seek to maximize the *expected information gain*, defined as the expected reduction in the Shannon entropy of our beliefs given a path-based sensor reading integrated over the distribution of possible events.

The contributions of our paper include:

- An iterative (recursive) Bayesian framework for path-based sensor readings. This can be used to refine beliefs about whether or not phenomena (hazards, hatches, etc.) exist at various locations in the environment given the output of a path-based sensor.
- We show how to calculate the expected Shannon information gain that will result from sending an agent along a *particular* path for both the Ex 1,2 and Ex 3 problems.
- We present a number of algorithms to solve problems Ex 1 and Ex 2, and Ex 3.
- We perform experiments in simulation comparing our methods to other ideas from the literature. While these previous ideas are not designed for use with path-based sensors, they are the most reasonable existing ideas that can be applied to this problem.

Our general mathematical formulations are valid in both continuous and discrete models of the environment. However,

for practical reasons of implementation, the algorithms that we present and the experiments that we run use a discrete *belief map* cellular decomposition of the environment. The world is modeled as a collection of non-overlapping cells, and we track our belief that each cell contains a hazard (and/or target), and paths are designed to maximize the expected information that is gained over all cells that a path traverses.

This paper is concerned with scenarios in which information is gathered by one robot at a time. While multiple robots may be used to enable subsequent paths to depart from or end at different locations in the environment, we assume the paths are traversed one at a time. The algorithms that we present for calculating the next robot's path fall into two categories: (1) optimal algorithms, which we find to be inefficient due to the general sub-modularity of the information gathering problem, and (2) efficient algorithms that are not optimal but that are shown to perform comparably to the optimal algorithms in our experiments.

Our formulations follow Bayesian and Shannon formulations. Our only departure from a "purely" Bayesian/Shannon formulation is that a user may opt to assign a different relative weight to each type of information that is collected, e.g., reflecting the fact that the user may desire more or less Shannon information about hazards than targets.

The rest of this paper is organized as follows: Related Work appears in Section II. Nomenclature is introduced in Section III. The recursive Bayesian update for the path-based sensor and the information theoretic calculations are described in Section IV. Path-based sensor information gathering problems are formally defined in Section V. Algorithms to solve the path-based sensor information gathering problems are described in Section VI. Runtime analysis is in Section VII. In Section VIII we run a number of experiments and discuss the results. Our conclusions are presented in Section IX. An appendix contains a simple example of a scenario in which the efficient algorithms that we present produce a sub-optimal solution, mid-level optimizations that we use to make the algorithms run quickly, a relaxed version of our method that runs in polynomial time, and additional experiment figures.

II. Related Work

The main difference between our work and all previous work is that we consider problems involving the question: "did an event occur somewhere along the robot's path?" Different bodies of related work are discussed in Sections II-A through II-H. The most closely related work is discussed first and our own previous work is discussed last. Large surveys of target search can be found in [3] and [4].

A. Closely Related Work

The approach presented in [5] uses recursive Bayesian filters to update estimates of both target and hazard locations. The location dependent probability of agent destruction is used to calculate the mutual information that can be gained about *targets*. Unlike the problem we consider, [5] assumes that agent failure locations are directly observable.

B. Related Work on Information Gathering

The first formal derivation of the gradient of mutual information appears in [6], which also proves that a multi-agent control policy of gradient ascent will gather all information, in the limit, as time goes to infinity. A multi-agent problem in which robots have full knowledge of positions and sensor data is posed, and a more tractable 1-step look ahead method, "information surfing," is used to solve problems involving intermittent communication between robots.

1-step look-ahead information surfing is also used in [7] for multi-agent target-localization, extended in [8] by implementation on a quad rotor test-bed, and extended in [9] to 3-step receding horizon approach use with rovers. Our work differs from [6]–[9] in that we consider a problem variant in which hazards can destroy agents.

Receding horizon control for gathering information about randomly moving targets is presented in [10], [11]. A multi-agent control framework to track moving targets is used in [10], while communication limitations to a receiver tower while tracking a single target is considered in [11]. Our work differs in that we consider path-based sensors, agent loss, and environments without communication.

An overview of grid-based environmental representations for information gathering appears in [12]. Approaches can be divided based on what form of data is stored in the grid. For example, a grid-based probability density function of a particular target's location is stored in [13], while the probability that each cell in a grid contains a particular phenomenon (target, environment class, etc.) is tracked in [14]–[16]. Our work differs from [13]–[16] in that we consider path-based sensors, and situations that may involve robot failure.

C. Other Related Work Involving Bayesian Search

A variety of Bayesian approaches have been used for target search and tracking. One approach is to track each target individually, for example using a Kalman filter. Each robot stores a vector of the targets that it has discovered so far in [17]. An alternative approach is to track the probabilities that map grid cells containing a target [18]. The resulting data structure is called a probabilistic map. Such grid-based approaches usually store a probabilistic belief at each cell, and then update these beliefs using Bayesian inference.

At least two different types of probabilistic maps exist: in Type 1 the grid stores different parts of a numerical probability density function of a single target's location [10], [19]–[27]; in Type 2 each grid tracks the probability that a target is located within the space that the grid represents [8], [9], [28]–[32].

We use Type 2 probabilistic maps because we find them convenient for locating multiple stationary targets/hazards. In problems involving both targets and hazards, we use one map for targets and another for hazards. Our work is the first to consider an iterative Bayesian update for a path-based sensor.

D. Planning for Submodular Objectives

Information gathering involves optimizing a submodular function. A general submodularity relaxation, called α-submodularity, which permits bounded results assuming a

suitable heuristic exists is presented in [33]. Submodularity has also been studied in the context of multi-robot search problems; a "discount function" over a finite time horizon that is defined as a decreasing function of the probability that other vehicles also decide to search a map cell is used in [34]. A finite time horizon search method to determine the multi-path for a robot team assuming other vehicles are treated as stochastic elements is used in [30], [35], [36]. Similar ideas are used to solve the related problem of visiting points of interest while avoiding threats [37].

E. Quasi-Probabilistic Methods

A number of methods combine Bayesian and/or frequentist ideas with non-probabilistic techniques to calculate "probability" values using methods that are not formally probabilistic or statistical in nature [38]–[41]. Proxies for probability values are also used to provide a notion of the relative uncertainty that exists with respect to different parts of the environment [42]–[50].

F. Non-Information-Theoretic and Non-Bayesian Search

Given a perfect sensor, a uniform prior over the location of the target(s), and unlimited fuel, a popular solution is to perform a lawn-mower sweep [51]. This idea has been extended to space-filling curves [52]. The robustness of multi-agent lawn-mower sweep to agent loss is investigated in [53], and replanning in response to missed observations is considered in [54]. Our work differs because we assume noisy sensors, start with a prior belief, and consider fuel constraints.

G. Related Work that Considers Hazards

A number of previous works have considered the risk of agent loss within a greater search (or target tracking) task. Vehicle loss risk is part of the cost or objective function in [36], [55]–[58] and hazards are treated as obstacles in [53], [59]. Our work differ from [36], [55], [56], [58] in that we seek to determine the locations of hazards using a path-based sensor, and from [53], [57], [59] in that we seek to maximize the information that is collected regarding our belief of hazard existence, and we do not have access to direct hazard observations.

H. Differences oo Our Previous Work

An initial version of this paper was presented at the Workshop on the Algorithmic Foundations of Robotics (WAFR) in 2019 [60]. Additional material in this journal version includes:

- A formal presentation of an algorithm that calculates the optimal path. This is different from the main algorithm that has already been presented in [60].
- A relaxation of the method from [60] is presented (see appendix). This new variation has a polynomial worst case runtime, while the method from [60] is, in the worst case, exponential in the number of repeated visits to the most visited cell.
- A formal analysis of the worst case runtime of our original method from [60], its new relaxation, and the optimal method.
- Additional experiments comparing the efficiency and performance of all three algorithms to each other and to other ideas. One experiment shows that the optimal solution cannot reasonably scale beyond short path lengths. Another experiment shows how false positive and false negative rates affect performance. A toy example shows how fifteen paths each contribute to changes in the entropy and beliefs about the environment.
- We have included the low-level implementation details of the dynamic programming subroutine used to efficiently calculate the information integral used in the optimal solution as well as the original solution proposed in [60]. Although not strictly necessary to reproduce the basic idea from [60], the dynamic programming solution is necessary to implement the idea efficiently.
- The paths calculated by our method [60] work well but are not guaranteed to be optimal. An example of a suboptimal solution is included in the appendix.
- This related work section contains a more comprehensive survey than that in [60].

In other prior work [61] we investigate how the existences of symmetries in the search space can be exploited to increase the efficiency of informed path planning algorithms, assuming a standard sensor and target search problem. We consider a target search game in [62] where two teams compete to locate a target using sweep sensors and evaluate the effects of communication on the probability of winning this game.

III. NOMENCLATURE

The search space is denoted $\mathbf{S}$ and is assumed to be a subset of two dimensional Euclidean space $\mathbf{S} \subset \mathbb{R}^2$. The vector $s \in \mathbf{S}$ describes a point in $\mathbf{S}$. We are provided a set W of k fixed uplink sites, $W = \{w_1, \ldots, w_k\} \subset \mathbf{S}$ where agents can upload the data they collect and download data collected by other agents to/from a remote database. We shall sometimes denote uplink points as 'start' and 'goal.'

A. Quantities Relevant To Path-Based Sensors

A path ζ is a mapping from the interval $[0, 1]$ to the state space $\mathbf{S}$. We will assume that paths are piecewise continuous curves that start and end at uplink sites. Formally, $\zeta : [0, 1] \rightarrow \mathbf{S}$ such that $\zeta(0) = s_{\text{start}} = w_a$ and $\zeta(1) = s_{\text{goal}} = w_b$ for $w_a, w_b \in W$. We allow both $w_a = w_b$ and $w_a \neq w_b$. With an abuse of notation to improve the overall clarity of our presentation, we overload the symbol ζ to additionally represent the set of points contained in the path that it defines, i.e., we can also say $\zeta \subset \mathbf{S}$.

We use Z to denote the state of the environment with respect to the phenomenon we are gathering information about using a path-based sensor. Formally, Z is a discrete time random variable that takes values on alphabet $\mathcal{Z}$. Let Q be an observation at a location in the environment; e.g, a robot failure in the case we are gathering information about hazards or the collection of a successful insect specimen collection if

we are looking for a nest. Q takes values on alphabet $\mathcal{Q}$. Even though we *cannot observe* the specific location in $\mathbf{S}$ where these events occurs, we know that they happen *somewhere along the path*. Assuming the environment is large and fuel is limited, robots that follow a particular path encounter a strict subset of the environment in a particular order. Defining Q is useful because it helps us to reason about where along a path an event has occurred.

In our algorithms and experiments we assume the environment is discretized such that paths are broken into a finite number of edges between nodes. For the purpose of inferring where an event took place, the traversal of path segments from node to node is modeled by a global discrete time counter that uses the variable $\tau = 1, 2, 3 \ldots$.

The path-based sensor may produce false positives p_{FalsePos} and/or false negatives p_{FalseNeg}. To accurately model the effect that path length has on false positives, it is convenient to define p_{FalsePos} and p_{FalseNeg} per time step, so that the probability of a false positive or false negative is considered separately along each path segment. Assuming that the event has occurred during a particular time step, then the probability of observing a true positive during that time step is given by the complementary probability $1 - p_{\text{FalseNeg}}$. We assume that false positives and false negatives occur with a known probabilities p_{malfunc} and p_{kill} per time step, respectively, such that $0 \leq p_{\text{malfunc}} < 1$ and $p_{\text{FalsePos}} \equiv p_{\text{malfunc}}$ and $0 < p_{\text{kill}} \leq 1$, and $p_{\text{FalseNeg}} \equiv 1 - p_{\text{kill}}$.

In the hazard detection example (Ex 2) false positives occur when the agent malfunctions for reasons unrelated to hazards. False negatives occur when a hazard does not destroy an agent that has entered its cell. We assume that if a hazard fails to destroy an agent, then the hazard remains unobserved.

Given the possibility of false positives and false negatives, the output observation of the path-based sensor is itself a random event that depends on both the path taken, and the presence or absence of the event triggering phenomenon (e.g., insect nests) in the environment. Let $\theta_{\zeta,1}$ and $\theta_{\zeta,0}$ denote the complementary events that the path-based sensor is triggered *somewhere* along ζ or is *not* triggered *anywhere* along ζ, respectively. We use 1 and 0 in the subscript to denote "trigger" and "not trigger", respectively. Let Θ_ζ be the random variable associated with survival of a path ζ. In general, the probability of these events is defined by a functional that accounts for motion along the path, and the presence of the environmental phenomenon. $\mathbb{P}(\Theta_\zeta = \theta_{\zeta,0}|Z) = f(\zeta, Z)$ and $\mathbb{P}(\Theta_\zeta = \theta_{\zeta,1}|Z) = 1 - \mathbb{P}(\Theta_\zeta = \theta_{\zeta,0}|Z)$. The particular form of f depends on the way that the environment is modeled.

As an example, we now restate the previous paragraph using values relevant to our running example of hazard search (Ex 2). $\theta_{\zeta,\text{alive}} \equiv \theta_{\zeta,0}$ and $\theta_{\zeta,\text{dead}} \equiv \theta_{\zeta,1}$ denote the complementary events that the robot survives the path ζ or does not, respectively. Θ_ζ is the random variable associated with survival of a path ζ. Thus, $\mathbb{P}(\Theta_\zeta = \theta_{\zeta,\text{alive}}|Z) = f(\zeta, Z)$ and $\mathbb{P}(\Theta_\zeta = \theta_{\zeta,\text{dead}}|Z) = 1 - \mathbb{P}(\Theta_\zeta = \theta_{\zeta,\text{alive}}|Z)$.

B. Quantities Relevant to Typical Sensors

In addition to collecting information about a particular phenomenon using a path-based sensor, we may wish to simultaneously gather additional information about a secondary phenomenon using a standard sensor. In the case where the primary phenomenon is environmental hazard presence and the robot cannot communicate from the field, the formulation requires consideration of the fact that the probability of robot destruction $\mathbb{P}(\Theta_\zeta = \theta_{\zeta,\text{dead}}|Z)$ directly affects the expected information that will be gathered about the secondary phenomenon.

Let X denote the state of the environment with respect to the secondary phenomenon, e.g., target presence. In general, X is a discrete time random variable that takes values on alphabet $\mathcal{X}$. In the stationary target scenario we consider, X is constant over time. Y is a sensor observation (also a random variable) of a portion of the environment, and takes values on an alphabet $\mathcal{Y}$. We assume that environmental sensor measurements occur at discrete times and are indexed by the variable $t = 1, 2, 3 \ldots$. Target sensor measurements may happen independently of path segment traversals such that $\tau \neq t$, in general. For example, if we use a different cellular decomposition of the same environment to reason about targets and hazards, respectively. The methods that we present are able to track τ and t independently; however, for simplicity we assume $\tau = t$ in our algorithmic presentation and experiments.

The robot is assumed to take sensor measurements about a secondary phenomenon using a standard sensor as it travels along the path. We consider the discrete case where one observation is made at each node in the path. Given this assumption, and assuming that t measurements have already been taken *before* an agent starts moving along its path, then the successful completion of a path provides an ordered finite set of sensor observations $\{y_{t+1}, \ldots, y_{t+\ell}\}$, where y_k is taken at position $s_k \in \zeta$ and $t+1 \leq k \leq t+\ell$, where ℓ is the number of sensor readings taken along the path ζ.

C. Additional Notation Used In the Algorithmic Presentation

The environment is modeled as discrete map $\mathbf{M}$ of non-overlapping cells $M_i \subset \mathbf{M}$, where $1 \leq i \leq m$ and $M_i \cap M_j = \emptyset$ for $i \neq j$ (see Figure 2). Target and hazard effects are assumed to be local to the map cells containing those targets and hazards, respectively. These assumptions are useful in practice because they reduce computational complexity. To simplify our presentation the same map $\mathbf{M} = \bigcup_{i \in [1,m]} M_i$ is used to reason about both targets and hazards. Because target and hazard effects are local to each cell, our beliefs about targets and hazards are stored in arrays $\mathbf{X}$ and $\mathbf{Z}$, where $\mathbf{X}[i]$ and $\mathbf{Z}[i]$ are our current beliefs that map cell M_i contains a target[1] and a hazard, respectively. Connectivity information is stored in a graph $G_\mathbf{S} = (V_\mathbf{S}, E_\mathbf{S})$. Each map cell M_i has

[1]In the most general (and intractable) discrete formulation of the ideas presented in Sections III-V target existence across all cells in the map is represented by a single random variable X that takes one of the 2^m different possible values x, depending on which cells contain targets and which do not. The set of all 2^m possibilities forms the alphabet $\mathcal{X}$. However, if each target only affects target sensor readings in its own cell, as we consider in this paper, then the resulting independence between cells allows us to consider each of the m dimensions of X separately. In other words, we can consider X as a joint event over a collection of independent random variables $X_1, \ldots, X_m$ because $\mathbb{P}(X = x) = \prod_{i=1}^{m} \mathbb{P}(X_i = x_i)$. We store our current estimate of $\mathbb{P}(X_i = x_i)$ in $\mathbf{X}[i]$.

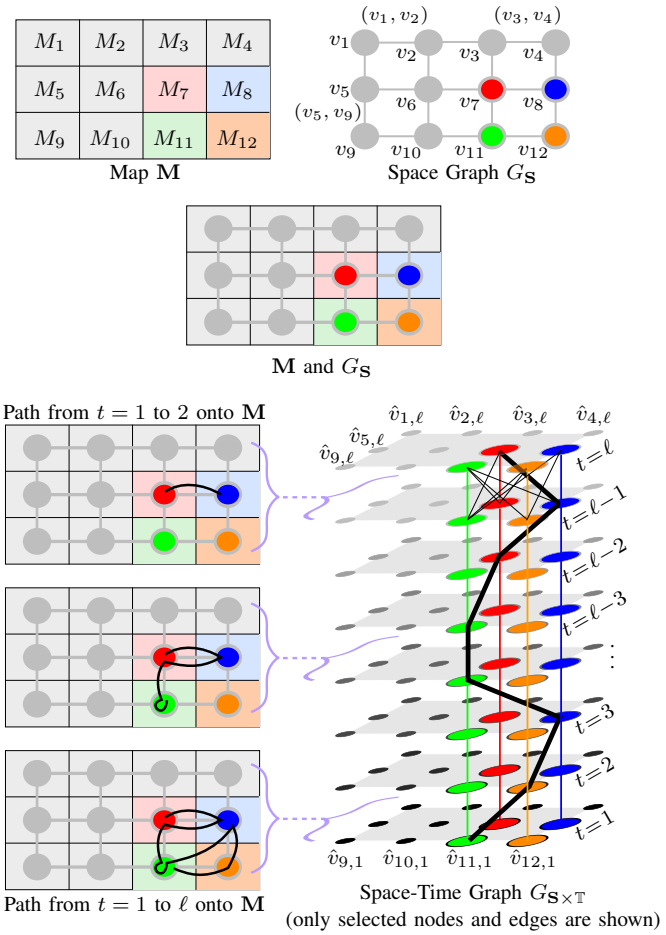

Fig. 2: Selected quantities used in the map $\mathbf{M}$, spatial graph $G_{\mathbf{S}} = (V_{\mathbf{S}}, E_{\mathbf{S}})$, and space-time graph $G_{\mathbf{S}\times\mathbb{T}} = (V_{\mathbf{S}\times\mathbb{T}\times\mathbb{H}}, E_{\mathbf{S}\times\mathbb{T}})$ that appear in our model of the environment. Each map cell M_i has a corresponding node $v_i \in V_{\mathbf{S}}$, and an edge $(v_i, v_j) \in E_{\mathbf{S}}$ indicates it is possible to move directly between map cells M_i and M_j. Self transitions $(v_i, v_i) \in E_{\mathbf{S}}$ are allowed, but can be removed in cases where agents must remain in motion. Edges in $E_{\mathbf{S}\times\mathbb{T}}$ move forward in time, and exist according to the following rule: $(v_i, v_j) \in E_{\mathbf{S}} \implies (\hat{v}_{i,t-1}, \hat{v}_{j,t}) \in E_{\mathbf{S}\times\mathbb{T}}$ for all $t \in [1, \ell]$. Nodes corresponding to four map cells are shown in four different colors, respectively.

a corresponding node $v_i \in V_{\mathbf{S}}$, and an edge $(v_i, v_j) \in E_{\mathbf{S}}$ indicates it is possible to move directly between map cells M_i and M_j. Self transitions $(v_i, v_i) \in E_{\mathbf{S}}$ are allowed, but can be removed in cases where agents must remain in motion.

Mutual information is sub-modular—there are diminishing returns for visiting the same cell again and again. Therefore, we plan in $G_{\mathbf{S}\times\mathbb{T}} = (V_{\mathbf{S}\times\mathbb{T}\times\mathbb{H}}, E_{\mathbf{S}\times\mathbb{T}})$ the space-time extension of $G_{\mathbf{S}}$, to track cell visit counts along a path. Agents have enough fuel for ℓ moves, so $G_{\mathbf{S}\times\mathbb{T}}$ is created by placing a "clone" $V_{\mathbf{S},t} \equiv V_{\mathbf{S}}$ at each of the $0 \leq t \leq \ell + 1$ time steps that must be considered, i.e., $V_{\mathbf{S}\times\mathbb{T}\times\mathbb{H}} = V_{\mathbf{S},0} \cup \ldots \cup V_{\mathbf{S},\ell}$. Edges in $E_{\mathbf{S}\times\mathbb{T}}$ move forward in time, and exist according to the following rule: $(v_i, v_j) \in E_{\mathbf{S}} \implies (\hat{v}_{i,t-1}, \hat{v}_{j,t}) \in E_{\mathbf{S}\times\mathbb{T}}$ for all $t \in [1, \ell]$. A valid path ζ_{valid} is a sequence of edges that starts at some uplink site $w_{start} = \hat{v}_{j,0}$ at time $t = 0$ and moves from node to node along edges in space-time until reaching a (goal) uplink site $w_{goal} = \hat{v}_{j,\ell}$ at time $t = \ell$.

If β is a belief that one of two complementary events has occurred, its entropy is calculated: $H(\beta) = -(\beta \log(\beta) + (1-\beta)\log(1-\beta))$. Given our assumption of cell independence, the total entropy regarding targets is $H(\mathbf{X}) = \sum_{i=1}^{m} H(\mathbf{X}[i])$ and total entropy regarding hazards is $H(\mathbf{Z}) = \sum_{i=1}^{m} H(\mathbf{Z}[i])$. Let $p_\zeta^{\text{alive}} \equiv \mathbb{P}(\theta_{\zeta,\text{alive}})$ and $p_\zeta^{\text{dead}} \equiv \mathbb{P}(\theta_{\zeta,\text{dead}})$.

We use the shorthand $\mathbf{0}_{1:k}$ to denote a one dimensional vector of length k in which every value is 0. For example $\mathbf{0}_{1:3} = [0, 0, 0]$.

The quantity $\zeta_{\hat{v}}$ contains the current subpath, the best that we have yet found, going from $\hat{v}$ to a goal. We store these quantities with each node.

IV. Bayesian Belief Updates And Expected Information Gain

We shall now derive a recursive Bayesian update for the path-based sensor that assumes a discrete time model. This model supports continuous paths, assuming any path can be partitioned into a finite number ℓ of path segments in order to reason about the various map cells it traverses. There are two cases to consider, which we now describe at a high level. Our presentation in this section uses the language related to the hazard tracking example (Ex 2).

Case 1, the path-based sensor is not tripped: If the path-based sensor is not tripped, then we can directly update our belief map based on the fact that negative hazard observation has taken place along each of the ℓ segments in the path. Each of the ℓ updates follows the well-known iterative Bayesian update for a typical sensor.

Case 2, the path-based sensor is tripped: When the path-based sensor is tripped a belief update is possible by considering separately each possibility—the finite set of mutually exclusive events that the sensor was tripped while traveling along each path segment $k \in [1, \ell]$—and then combining the resulting ℓ separate belief maps weighted by the relative likelihood of each occurring. This is done by taking the results of all previous measurements as a prior on the current path traversal, and then accounting for the probability of false positives and false negatives along the path. The individual update for the possibility that the sensor was tripped on segment $k \in [1, \ell]$ is simply the standard Bayesian update that would be used *if we knew for certain* that the agent had been destroyed along segment k. Given the set of all possibilities, and their relative likelihoods, we then use the standard Bayesian trick of re-normalization so that the total probability mass sums to one. The resulting entropy can be calculated using the same general idea.

The remainder of this section focuses on the mathematical formulations of the Bayesian update rules and information theoretic calculations of both cases. The mathematical formulation of a standard recursive Bayesian update (such as for a camera) is presented in Section IV-A and used as a subroutine for the path-based sensor updates. The more involved description of Case-2 is also presented in Section IV-B. A standard sensor update is also used when we consider a secondary environmental phenomenon of interest for both positive and

negative sensor readings. The presentation in IV-B is described using the X, Y, and t notation. The update case for each of the multiple segment updates that occur when the path-based sensor is not tripped (Case 1) is obtained by replacing these quantities with Z, Q, and τ, respectively.

A. Typical Sensor Updates

Let X_0 denote the prior belief defined over $\mathbf{S}$ that each point $s \in \mathbf{S}$ contains the phenomenon in question, e.g., a target. For notational convenience, we increment the time index t based on the number of successfully communicated sensor measurements; i.e., t ordered sensor observations have been delivered to the uplink points by time t. Given X_0 and sensor measurements $y_1, \ldots, y_t$, which may have been taken across multiple paths of varying lengths, an iterative Bayesian update can be used to compute $\mathbb{P}(X_t | y_1, \ldots, y_t)$, the posterior probability of X given the t sensor readings delivered to the uplink points by time t.

$$\begin{aligned}&\mathbb{P}(X_t | Y_1 = y_1, \ldots, Y_t = y_t) \\ &= \frac{\mathbb{P}(Y_t = y_t | X_{t-1}) \, \mathbb{P}(X_{t-1} | Y_1 = y_1, \ldots, Y_{t-1} = y_{t-1})}{\mathbb{P}(Y_t = y_t | Y_1 = y_1, \ldots, Y_{t-1} = y_{t-1})}\end{aligned} \tag{1}$$

The denominator need not be explicitly calculated; as is often done, we calculate the numerators of Equation 1 for all events $X_t = x \in \mathcal{X}$ and then normalize such that they sum to 1.

The information entropy of X_t is denoted $H(X_t)$ and defined:

$$H(X_t) = -\int_{x \in \mathcal{X}} \mathbb{P}(X_t) \log \mathbb{P}(X_t) \, dx$$

and provides a measure of the unpredictability of X_t. As entropy increases, X_t is essentially "worse" at being able to predict the presence or absence of the phenomenon. In other words, its values are closer to a uniformly random process.

The conditional information entropy $H(X_{t+1} | Y_{t+1})$ is the updated entropy of the environmental state X given a new observation Y_{t+1}, *averaged over all possible values that* Y_{t+1} *may take*. The difference between the entropy $H(X_t)$ and the conditional entropy $H(X_{t+1} | Y_{t+1})$ is called *mutual information*, defined $I(X_t; Y_{t+1}) = H(X_t) - H(X_{t+1} | Y_{t+1})$. Mutual information quantifies the *expected* reduction in the unpredictability of our estimation of X given the new measurement Y_{t+1}.

It is useful to calculate the mutual information of a standard sensor measurement Y_{t+1} before it is taken, so that we may compare the expected benefits of sampling various locations. The mutual information of a new observation is calculated:

$$\begin{aligned}I(X_t; Y_{t+1}) = \int_{y \in \mathcal{Y}} \int_{x \in \mathcal{X}} &\mathbb{P}(Y_{t+1} = y, X_t = x) \\ &\cdot \log \left(\frac{\mathbb{P}(Y_{t+1} = y, X_t = x)}{\mathbb{P}(Y_{t+1} = y) \, \mathbb{P}(X_t = x)} \right) dx dy\end{aligned}$$

where

$$\mathbb{P}(Y_{t+1} = y, X_t = x) = \mathbb{P}(Y_{t+1} = y | X_t = x) \, \mathbb{P}(X_t = x).$$

We desire paths that gather as much mutual information as possible, given fuel constraints and other goals. Given a path ζ that enables sensor observations $y_{t+1}, \ldots, y_{t+\ell}$ if and only if it is completed successfully, the expected cumulative information gained along that path, given all measurements so far, is calculated:

$$I(X_t; Y_{t+1}, \ldots, Y_{t+\ell}) = \sum_{k=t+1}^{t+\ell} I(X_{k-1}; Y_k | Y_{t+1}, \ldots, Y_{k-1})$$

where the notation $I(A; B|C)$ denotes the conditional mutual information of A and B, integrated over all possible outcomes in the event space of C and weighted by their relative likelihoods. That is, $I(A; B|C) = \mathbb{E}_C \left(I(A; B) | C \right)$.

In the most general case, in which a phenomenon at any location in the environment may affect sensor readings at any other location, the calculation of $I(X_t; Y_{t+1}, \ldots, Y_{t+\ell})$ can become intractable because the number of terms involved in the computation of the inner $I(X_{k-1}; Y_k | Y_{t+1}, \ldots, Y_{k-1})$ scales according to $|\mathcal{Y}|^k$. However, this complexity can be reduced, e.g., to a small constant, by assuming that a phenomenon only affects sensor observations in its own local neighborhood.

In Ex 3 no information about the secondary phenomenon is actually gained, from the point-of-view of the planning system, until the robot reaches an uplink point. Hence, *no information about* ***targets*** *is gathered in the event that the robot is destroyed along its path.* In such a scenario the expected mutual information along a particular path, assuming the robot may or may not be destroyed along that path, is:

$$\begin{aligned}&I(X_t; Y_{t+1}, \ldots, Y_{t+\ell} | \Theta_\zeta) \\ &\qquad = \mathbb{P}(\Theta_\zeta = \theta_{\zeta,\text{alive}}) \, I(X_t; Y_{t+1}, \ldots, Y_{t+\ell}).\end{aligned}$$

B. Path-Based Sensor Updates

Both events $\Theta_\zeta = \theta_{\zeta,0}$ and $\Theta_\zeta = \theta_{\zeta,1}$ can be used to perform an iterative Bayesian update of Z based on ζ. However, the iterative updates to Z based on Θ_ζ take different forms depending on if $\Theta_\zeta = \theta_{\zeta,0}$ or $\Theta_\zeta = \theta_{\zeta,1}$. In other words, in hazard detection, different updates are used depending on if $\Theta_\zeta = \theta_{\zeta,\text{alive}}$ or $\Theta_\zeta = \theta_{\zeta,\text{dead}}$, respectively.

We begin by noting that *if* we somehow had access to the direct observations at all cells along the path, then a straightforward belief update would be the following:

$$\begin{aligned}&\mathbb{P}(Z_{\tau+j} | Z_\tau, Q_{\tau+1} = q_{\tau+1}, \ldots, Q_{\tau+j-1} = q_{\tau+j-1}) \\ &= \mathbb{P}(Q_{\tau+j} = q_{\tau+j} | Z_{\tau+j-1}) \\ &\quad \cdot \frac{\mathbb{P}(Z_{\tau+j-1} | Z_\tau, Q_{\tau+1} = q_{\tau+1}, \ldots, Q_{\tau+j-1} = q_{\tau+j-1})}{\mathbb{P}(Q_\tau = q_\tau | Z_\tau, Q_{\tau+1} = q_{\tau+1}, \ldots, Q_{\tau+j-1} = q_{\tau+j-1})}\end{aligned} \tag{2}$$

Next, we observe that whenever the sensor is *not* tripped we *do* have direct access to all "observations" of hazards along the path. All observations are either true negatives or false negatives, since the path-based sensor was not tripped, and so they are $Q_j = q_j = 0$ by construction. Formally, $\Theta_\zeta = \theta_{\zeta,0} \iff q_{\tau+1} = 0, \ldots, q_{\tau+l} = 0$. Thus, we simply perform the standard update:

$$\begin{aligned}&\mathbb{P}(Z_{\tau+j} | \Theta_\zeta = \theta_{\zeta,0}) \\ &\qquad = \mathbb{P}(Z_{\tau+l} | Z_\tau, Q_{\tau+1} = 0, \ldots, Q_{\tau+l} = 0)\end{aligned}$$

which can be computed iteratively, for each $j = 1, \ldots, l$ (by repeatedly applying the typical Bayesian update described in the previous section) as follows:

$$\begin{aligned}&\mathbb{P}\left(Z_{\tau+j} | Z_\tau, Q_{\tau+1} = 0, \ldots, Q_{\tau+j-1} = 0\right) \\ &= \mathbb{P}\left(Q_{\tau+j} = 0 | Z_{\tau+j-1}\right) \\ &\qquad \cdot \frac{\mathbb{P}\left(Z_{\tau+j-1} | Z_\tau, Q_{\tau+1} = 0, \ldots, Q_{\tau+j-1} = 0\right)}{\mathbb{P}\left(Q_\tau = 0 | Z_\tau, Q_{\tau+1} = 0, \ldots, Q_{\tau+j-1} = 0\right)}\end{aligned}$$

We now move onto the case that the path-based sensor is tripped $\Theta_\zeta = \theta_{\zeta,1}$ or, in the hazard example, the agent does not survive $\Theta_\zeta = \theta_{\zeta,\text{dead}}$. The recursive Bayesian update of Z must take a different form. Given a path with l segments, with the first segment starting at time τ, the event $Q_{\tau+j} = 1$ is equivalent to the statement "the path-based sensor was tripped along the j-th segment of the path."

Given $\Theta_\zeta = \theta_{\zeta,1}$, we know that the path-based sensor was tripped *somewhere* along ζ, but we do not know where. However, we can integrate over all l possibilities, i.e., considering each possibility that it was tripped on a different path segment j for all j such that $1 \leq j \leq l$, and then summing these results weighted by the relative probability of each given our current hazard beliefs.

In order to build intuition, it is convenient to use the metaphor of a multiverse. We simultaneously assume the existence of j different universes, such that in the j-th universe the sensor was tripped along the j-th path segment. Assuming we are in a particular j-th universe, we can calculate the iterative Bayesian update to Z by applying Equation 2 exactly j times, assuming that on the k-th application:

$$Q_{\tau+k} = q_{\tau+k} = \begin{cases} 0 & \text{if } k < j \\ 1 & \text{if } k = j \end{cases}$$

and where no observations are made for $k > j$ in the j-th universe. Let $Z^j_{\tau+l}$ denote the version of $Z_{\tau+l}$ that is calculated in the j-th universe.

The final overall update to the "real" $Z_{\tau+l}$ is the expected value of $Z_{\tau+l}$ in the multiverse, found by combining all $Z_{\tau+l}$ weighted by $\mathbb{P}\left(Q_{\tau+j} = 1 | Z_\tau, \Theta_\zeta = \theta_{\zeta,1}\right)$, the probability of being in the j-th universe of the multiverse.

$$Z_{\tau+l} = \sum_{j=1}^{l} \mathbb{P}\left(Q_{\tau+j} = 1 | Z_\tau, \Theta_\zeta = \theta_{\zeta,1}\right) Z^j_{\tau+l} \tag{3}$$

The quantity $\mathbb{P}\left(Q_{\tau+j} = 1 | Z_\tau, \Theta_\zeta = \theta_{\zeta,1}\right)$ can be obtained by calculating the probability that the path-based sensor remains untripped to the j-th path segment given Z_τ, and is then tripped somewhere along the j-th path segment. After doing this for each of the path segments, we renormalize such that the total probability mass of all l possibilities sums to 1.

$$\begin{aligned}&\mathbb{P}\left(Q_{\tau+j} = 1, Z_\tau, \Theta_\zeta = \theta_{\zeta,1}\right) \\ &\quad = \frac{\mathbb{P}\left(Q_{\tau+j} = 1, Z_\tau\right) \prod_{k=1}^{j-1} \mathbb{P}\left(Q_{\tau+k} = 0, Z_\tau\right)}{\sum_{j=1}^{l} \mathbb{P}\left(Q_{\tau+j} = 1, Z_\tau\right) \prod_{k=1}^{j-1} \mathbb{P}\left(Q_{\tau+k} = 0, Z_\tau\right)}\end{aligned}$$

where

$$\begin{aligned}&\mathbb{P}\left(Q_{\tau+k} = q, Z_\tau\right) \\ &\qquad = \int_{z \in \mathcal{Z}} \mathbb{P}\left(Q_{\tau+k} = q | Z_\tau = z_\tau\right) \mathbb{P}\left(Z_\tau = z_\tau\right) dz\end{aligned}$$

for $q \in \{0, 1\}$.

The expected decrease in entropy about hazard locations gained from sending an agent along path ζ can be calculated by first calculating the conditional decrease in entropy assuming either possibility of $\Theta_\zeta = \theta_{\zeta,0}$ and $\Theta_\zeta = \theta_{\zeta,1}$ independently, and then combining the results weighted by the probability of each event given Z_τ.

Let $Z^{\theta_{\zeta,\text{alive}}}_{\tau+l}$ be the value of $Z_{\tau+l}$ that results if the path-based sensor remains untripped along the entire path—as calculated according to Equation 2. Similarly, let $Z^{\theta_{\zeta,1}}_{\tau+l}$ be the result if the sensor is tripped—as calculated by Equation 3.

The mutual information regarding hazards—the expected information gained from a path-based sensor observation—is given by the expected reduction in entropy: $I(Z_\tau; \Theta_\zeta) = H(Z_\tau) - H(Z_{\tau+l} | \Theta_\zeta, Z_\tau)$, where

$$H(Z_{\tau+l} | \Theta_\zeta, Z_\tau) = \int_\theta \mathbb{P}\left(\Theta_\zeta = \theta | Z_\tau\right) H(Z_{\tau+l} | \Theta_\zeta = \theta, Z_\tau) d\theta$$

is the conditional entropy, and

$$\mathbb{P}\left(\Theta_\zeta = \theta_{\zeta,0} | Z_\tau\right) = \prod_{j=1}^{l} \mathbb{P}\left(Q_{\tau+j} = 0, Z_\tau\right)$$

$$\mathbb{P}\left(\Theta_\zeta = \theta_{\zeta,1} | Z_\tau\right) = 1 - \mathbb{P}\left(\Theta_\zeta = \theta_{\zeta,0} | Z_\tau\right)$$

$$H(Z_{\tau+l} | \Theta_\zeta = \theta_{\zeta,0}, Z_\tau) = Z^{\theta_{\zeta,0}}_{\tau+l}$$

and

$$H(Z_{\tau+l} | \Theta_\zeta = \theta_{\zeta,1}, Z_\tau) = Z^{\theta_{\zeta,1}}_{\tau+l}.$$

V. Formal Problem Definitions

In this section we formally define the information gathering problems that we study in this paper, problems 1, 3, 4, and 5. For comparison, we also define the classic information gathering problem that has typically been studied in the past, problem 2. We use the triangle symbol '$\triangle$' to indicate the end of a formal problem statement.

The most basic path-based sensor information gathering problem involves designing a path in order to maximize the expected information that is gathered from the environment. Assuming we are given a subset of points at which the path can start/end, this problem is formally defined as follows:

Problem Definition 1. *Path-based sensor mutual information path planning*:

Given a search space $\mathbf{S}$, and a set of uplink points $\{w_1, \ldots, w_k\} = W \subset \mathbf{S}$, where an agent can start at any $w_{start} \in \mathbf{S}$ and end at any $w_{goal} \in \mathbf{S}$. Find the path ζ^* that maximizes the expected information gain about a phenomenon Z:

$$\zeta^* = I(Z_\tau; \Theta_\zeta)$$

where for all ζ it is true that $\zeta : [0, 1] \to \mathbf{S}$ s.t. $\zeta(0) = w_{start}$ and $\zeta(1) = w_{goal}$, subject to distance constraints $\|\zeta\| < \ell$, and

where Θ_ζ is the space of observations about whether or not the paths sensor makes an observation along the path. $\triangle$

It is useful to compare Problem 1 to the typical information gathering problem, where it is assumed that we have direct access to the sensor state at each time step along a path, and which is defined as follows:

Problem Definition 2. *Mutual information path planning (typical formulation for a non-path-based sensor)*:

Given a search space $\mathbf{S}$, and a set of uplink points $\{w_1, \ldots, w_k\} = W \subset \mathbf{S}$, where an agent can start at any $w_{start} \in \mathbf{S}$ and end at any $w_{goal} \in \mathbf{S}$, and assuming the agent has a noisy target sensor that provides observations Y about a phenomenon X each time step, find the path ζ^* that maximizes the expected information gain about that phenomenon X:

$$\zeta^* = \arg\max_{\zeta} I(X_t; Y_{t+1}, \ldots, Y_{t+\ell} | \Theta_\zeta)$$

where for all ζ it is true that $\zeta : [0,1] \rightarrow \mathbf{S}$ s.t. $\zeta(0) = w_{start}$ and $\zeta(1) = w_{goal}$, subject to distance constraints $\|\zeta\| < \ell$, and where Θ_ζ is the space of observations about whether or not the agent successfully completes the path. $\triangle$

Problem 2 is encountered, for example, when gathering information about search-and-rescue targets in the *absence* of hazards. Problem 2 differs from Problem 1 in that direct access to sensor data provides a sequence of separate measurements $Y_{t+1}, \ldots, Y_{t+\ell}$ instead of the single path-based measurement Θ_ζ that we get in Problem 1.

Problems 1 and 2 both involve a single path. Problem 3 is an iterative extension to Problem 1 in which robot(s) repeatedly follow paths, and we update our belief about the phenomenon we are monitoring after each path-based sensor reading.

Problem Definition 3. *Iterative path-based sensor mutual information path planning*:

Repeatedly solve Problem 1 to continually refine our belief about Z given Y and Θ_ζ, respectively. $\triangle$

In this general formulation, each path may start and end at any $w_{start}, w_{goal} \in \mathbf{S}$. However, if the same single agent is used across all paths—a modification only advisable in non-hazardous scenarios—then it is possible to add a requirement that w_{goal} of one path becomes the w_{start} of the next path.

Problem 4 formalizes the Ex 3 scenario, in which an agent operates in a hazardous environment while gathering information about a separate 'target' phenomenon.

Problem Definition 4. *Mutual information path planning for targets and hazards without communication*:

Given a search space $\mathbf{S}$, and a set of uplink points $\{w_1, \ldots, w_k\} = W \subset \mathbf{S}$, and assuming an agent can start at any $w_{start} \in \mathbf{S}$ and end at any $w_{goal} \in \mathbf{S}$, and assuming an agent has a noisy target sensor that provides observations Y about targets X, find the path ζ^* that maximizes the expected information gain about both targets X and hazards Z:

$$\zeta^* = \arg\max_{\zeta} c_X I(X_t; Y_{t+1}, \ldots, Y_{t+\ell} | \Theta_\zeta) + c_Z I(Z_\tau; \Theta_\zeta)$$

where $c_X, c_Z \in [0,1]$ are weights that represent user preference for either type of information, and where for all ζ it is true that $\zeta : [0,1] \rightarrow \mathbf{S}$ s.t. $\zeta(0) = w_{start}$ and $\zeta(1) = w_{goal}$, subject to distance constraints $\|\zeta\| < \ell$, and where Θ_ζ is the space of observations about whether or not the agent successfully completes the path. $\triangle$

We conclude this section by defining Problem 5 as the iterative extension to Problem 4.

Problem Definition 5. *Iterative mutual information path planning for targets and hazards without communication*:

Repeatedly solve Problem 4 to continually refine our belief about targets X and hazards Z given Y and Θ_ζ, respectively; assuming we are able to replace agents that are lost, once their failure to appear at their destinations has been observed, and that each path may start and end at any $w_{start}, w_{goal} \in \mathbf{S}$. $\triangle$

VI. Algorithms

In Section VI-A we present algorithms that solve Problems 1 and 3. In Section VI-B we present algorithms that solve Problems 4 and 5. In Section VI-C we describe the modifications necessary to create an optimal algorithm. Our presentation in this section uses language related to detecting hazards with a path-based sensor.

A. High Level Algorithms For Basic And Iterative Path-Based Sensor Information Planning (Problems 1 and 3)

The outer loop of the iterative planning approach that solves Problem 2 appears in Algorithm 1. We iteratively calculate the r-th path (line 2), send a robot along that path (line 3), observe if the robot survived (line 4), and then perform the appropriate Bayesian update depending if the robot survived (line 5) or did not (line 7). The update that occurs when the path-based sensor is not tripped is equivalent to recording a sequence of typical Bayesian updates assuming that we received a negative sensor reading ("0") at each node along the path. The three main subroutines that are used in Algorithm 5 appear in Algorithms 2-4, respectively.

The Subroutine `calculatePath`($\mathbf{Z}$) is described in Algorithm 2, and solves the basic iterative path-based sensor information path planning problem (Problem 1). It makes use of a first-in-first-out queue. A node $\hat{v}$ is inserted into the queue using `InsertFIFOQueue`($\hat{v}$) and the top node is removed from the queue using `PopFIFOQueue`. The subroutine `InQueue`($\hat{v}$) returns true if $\hat{v}$ is in the queue and otherwise false.

Algorithm 2 performs an efficient but suboptimal reverse search. (Optimal solutions suffer from a runtime that is exponential in path length, see Section VI-C.) Each node $\hat{v}_i$ has an associated quantity $h_{\hat{v}_i}$ which tracks the entropy expected to remain after we travel from that node to a goal. All $h_{\hat{v}_i}$

Algorithm 1 Iterative Path-Based Sensor Mutual Information Path Planning.
Input: Prior hazard beliefs $\mathbf{Z}$
Output: Iterative sequence of paths, and updates to $\mathbf{Z}$

1: **for** $r = 1, 2, \ldots$ **do**
2: $\quad\zeta =$ `calculatePath`($\mathbf{Z}$)
3: $\quad$Robot r moves along path ζ
4: $\quad$**if** $\theta_{\zeta,\text{alive}}$ **then**
5: $\quad\quad\mathbf{Z} \leftarrow$ `BayesianCellUpdates`($\mathbf{Z}, [0, \ldots, 0]$)
6: $\quad$**else**
7: $\quad\quad\mathbf{Z} \leftarrow$ `KilledOnPathUpdate`($\mathbf{Z}, \zeta$)

are initialized to negative infinite values (lines 1-2). We insert all of the goal nodes at time ℓ into the queue after initializing all of the associated subpaths to the empty set (lines 3-5). Next, via indirect use of the queue (lines 5, 6, 16, and 17), we consider all possible nodes at time ℓ, then their predecessors at time $\ell - 1$, etc. (lines 6-15). For each node $\hat{v}_i$ within time layer $\ell - 1$, we determine the best subpath that connects node $\hat{v}_i$ to a goal through one of its neighbors $\hat{v}_j$ (lines 6-14). This is done by considering all edges $(\hat{v}_i, \hat{v}_j)$ that move forward in time (toward the goal) from $\hat{v}_i$. (In practice, storing these with node $\hat{v}_j$ is advised.) The suboptimality of this algorithm is a direct consequence of considering *only* the set of paths that can be created by connecting nodes at time slice t to then best subpaths remembered at time $t + 1$, and not all of the longer subpaths that connect further forward in time.

We calculate the maximum amount of information that can be gained via this process by creating a temporary path $\zeta \leftarrow (\hat{v}_i, \hat{v}_j) + \zeta_{\hat{v}_j}$ (line 8) and then calculating the resulting expected entropy $\hat{h}_{this}$ that remains after traversing the entire path (lines 9-11). The lowest resulting entropy corresponds to the most information gain. The cases of destruction and survival are handled separately (lines 9 and 10, respectively) and then combined according to their probabilities of occurring (line 11)—the probability p_ζ^{alive} of surviving from $\hat{v}_i$ to the goal at the end of the path in question is calculated as part of the hypothetical destruction update (line 9). If this is the best predecessor path that we have found from $\hat{v}_i$ then we remember both the expected entropy and the subpath itself (lines 12-14).

We make sure that the predecessor nodes of $\hat{v}_i$ are in the queue, adding them if not, so that the process will eventually continue onto the next time slice (lines 15-17). Once the queue is empty, we select the start node for which we have found the most informative path to a goal (line 18), and then return that path (lines 19-20).

The subroutine $\texttt{KilledOnPathUpdate}(\mathbf{Z}, \zeta)$ in Algorithm 3 performs the Bayesian update that is run in the event that the robot is destroyed. This happens both as part of the outer iterative information search algorithm, and also as part of the calculation of the most informative subpath during the path planning process.

Algorithm 2 $\texttt{calculatePath}(\mathbf{Z})$
Input: Hazard beliefs $\mathbf{Z}$
Output: Path ζ

1: **for all** $\hat{v}_i \in V_{\mathbf{S}\times\mathbb{T}\times\mathbb{H}}$ **do**
2: $\quad h_{\hat{v}_i} = -\infty$
3: **for all** uplink points $w \in W_{\text{goal}}$ **do**
4: $\quad \zeta_w \leftarrow \emptyset$
5: $\quad \texttt{InsertFIFOQueue}(w)$
6: **while** $\hat{v}_i \leftarrow \texttt{PopFIFOQueue}$ **do**
7: $\quad$ **for all** $(\hat{v}_i, \hat{v}_j) \in E_{\mathbf{S}\times\mathbb{T}}$ **do**
8: $\quad\quad \zeta \leftarrow (\hat{v}_i, \hat{v}_j) + \zeta_{\hat{v}_j}$
9: $\quad\quad (\mathbf{Z}_{live}, p_\zeta^{\text{alive}}) \leftarrow \texttt{KilledOnPathUpdate}(\mathbf{Z}, \zeta)$
10: $\quad\quad \mathbf{Z}_{killed} \leftarrow \texttt{BayesianCellUpdates}(\mathbf{Z}, [0, \ldots, 0])$
11: $\quad\quad \hat{h}_{this} \leftarrow p_\zeta^{\text{alive}} H(\mathbf{Z}_{live}) + (1 - p_\zeta^{\text{alive}}) H(\mathbf{Z}_{killed})$
12: $\quad\quad$ **if** $\hat{h}_{this} > h_{\hat{v}_i}$ **then**
13: $\quad\quad\quad \zeta_{\hat{v}_i} \leftarrow \zeta$
14: $\quad\quad\quad h_{\hat{v}_i} \leftarrow h_{this}$
15: $\quad\quad$ **for all** $(\hat{v}_k, \hat{v}_i) \in E_{\mathbf{S}\times\mathbb{T}}$ **do**
16: $\quad\quad\quad$ **if not** $\texttt{InQueue}(\hat{v}_k)$ **then**
17: $\quad\quad\quad\quad \texttt{InsertFIFOQueue}(\hat{v}_k)$
18: $w_{start} \leftarrow \arg\min_{w \in W_{\text{start}}} h_w$
19: $\zeta \leftarrow \zeta_{w_{start}}$
20: **return** ζ

Algorithm 3 $\texttt{KilledOnPathUpdate}(\mathbf{Z}, \zeta)$
Input: Beliefs $\mathbf{Z}$, Path ζ which triggered path-based sensor
Output: Posterior Beliefs $\mathbf{Z}$

1: $p_1^{survivedTo} \leftarrow 1$
2: **for** $k \leftarrow 1, \ldots, \ell$ **do**
3: $\quad i \leftarrow$ index of cell in which k-th observation was made
4: $\quad p_k^{killedInGivenAt} \leftarrow (p_{\text{kill}} + p_{\text{malfunc}}(1 - p_{\text{kill}}))\mathbf{Z}[i] + p_{\text{malfunc}}(1 - \mathbf{Z}[i])$
5: $\quad p_{k+1}^{survivedTo} \leftarrow p_k^{survivedTo}(1 - p_k^{killedInGivenAt})$
6: $\quad \mathbf{Z}_k \leftarrow \mathbf{Z}$
7: $\quad \mathbf{Z}_k \leftarrow \texttt{BayesianCellUpdates}(\mathbf{Z}_k, [\mathbf{0}_{1:k-1}, 1])$
8: $p_\zeta^{\text{dead}} \leftarrow \sum_{k=1}^{\ell} p_k^{survivedTo}$
9: $\mathbf{Z} \leftarrow \sum_{k=1}^{\ell} \frac{p_k^{survivedTo}}{p_\zeta^{\text{dead}}} \mathbf{Z}_k$
10: **return** $(\mathbf{Z}, (1 - p_\zeta^{\text{dead}}))$

Algorithm 4 $\texttt{BayesianCellUpdates}(\mathbf{B}, \bar{\beta})$
Input: Beliefs $\mathbf{B}$, Standard sensor observation sequence $\bar{\beta}$
Output: Posterior beliefs $\mathbf{B}$

1: **for** $k = 0, \ldots, \ell$ **do**
2: $\quad i \leftarrow$ index of cell in which k-th observation was made
3: $\quad \mathbf{B}[i] \leftarrow \mathbb{P}\left(B_i \,|\, \mathbf{B}[i], \bar{\beta}[k]\right)$
4: **return** $\mathbf{B}$

Algorithm 5 Iterative information path planning for targets and hazards
Input: Prior target beliefs $\mathbf{X}$ and hazard beliefs $\mathbf{Z}$
Output: Iterative sequence of paths, and updates to $\mathbf{X}$ and $\mathbf{Z}$

1: **for** $r = 1, 2, \ldots$ **do**
2: $\quad \zeta = \texttt{calculatePath}(\mathbf{X}, \mathbf{Z})$
3: $\quad$ Robot r attempts path ζ
4: $\quad$ **if** $\theta_{\zeta,\text{alive}}$ **then**
5: $\quad\quad \mathbf{X} \leftarrow \texttt{BayesianCellUpdates}(\mathbf{X}, \mathbf{Y}_\zeta)$
6: $\quad\quad \mathbf{Z} \leftarrow \texttt{BayesianCellUpdates}(\mathbf{Z}, [0, \ldots, 0])$
7: $\quad$ **else**
8: $\quad\quad \mathbf{Z} \leftarrow \texttt{KilledOnPathUpdate}(\mathbf{Z})$

Algorithm 3 works by considering the probabilities that the agent survived to each time step (lines 2-7). This is calculated based on our previous estimate of hazard presence (lines 1-5). For each time step we record the hypothetical Bayesian update that would occur if we were certain that the agent was destroyed at that time (lines 6-7). Since the agent was destroyed, we renormalize the relative probabilities of each possibility so that the probability mass sums to one (line 8). Next, we combine all possible updates based on the relative probability they occurred (line 9). Finally, we return both the combined update as well as the probability, based on our previous beliefs, that the robot would have survived the entire path (line 10).

Subroutine $\texttt{BayesianCellUpdates}(\mathbf{B}, \bar{\beta})$ in Algorithm 4 shows the recursive Bayesian update that is used for the belief vector $\bar{\beta}$ given the observation vector B_i. Line 3 performs the recursive update that yields the posterior probability B_i regarding the i-th map cell.

Algorithm 6 `calculatePath`$(\mathbf{X}, \mathbf{Z})$
Input: Target beliefs $\mathbf{X}$ and hazard beliefs $\mathbf{Z}$
Output: Path ζ

1: **for all** $\hat{v}_i \in V_{\mathbf{S}\times\mathbb{T}\times\mathbb{H}}$ **do**
2: $h_{\hat{v}_i} = -\infty$
3: **for all** uplink points $w \in W_{\text{goal}}$ **do**
4: $\zeta_w \leftarrow \emptyset$
5: `InsertFIFOQueue`(w)
6: **while** $\hat{v}_i \leftarrow$ `PopFIFOQueue` **do**
7: **for all** $(\hat{v}_i, \hat{v}_j) \in E_{\mathbf{S}\times\mathbb{T}}$ **do**
8: $\zeta \leftarrow (\hat{v}_i, \hat{v}_j) + \zeta_{\hat{v}_j}$
9: $\hat{h}^{\mathcal{X}}_{live} \leftarrow \int_{x\in\mathcal{X}} H(\mathbf{X}_{live})dx$
10: $(\mathbf{Z}_{live}, p^{\text{alive}}_{\zeta}) \leftarrow$ `KilledOnPathUpdate`$(\mathbf{Z}, \zeta)$
11: $\mathbf{Z}_{killed} \leftarrow$ `BayesianCellUpdates`$(\mathbf{Z}, [0, \ldots, 0])$
12: $\hat{h}_{this} \leftarrow c_Z(p^{\text{alive}}_{\zeta} H(\mathbf{Z}_{live}) + (1 - p^{\text{alive}}_{\zeta})H(\mathbf{Z}_{killed})) + c_X p^{\text{alive}}_{\zeta} \hat{h}^{\mathcal{X}}_{live}$
13: **if** $\hat{h}_{this} > h_{\hat{v}_i}$ **then**
14: $\zeta_{\hat{v}_i} \leftarrow \zeta$
15: $h_{\hat{v}_i} \leftarrow h_{this}$
16: **for all** $(\hat{v}_k, \hat{v}_i) \in E_{\mathbf{S}\times\mathbb{T}}$ **do**
17: **if not** `InQueue`$(\hat{v}_k)$ **then**
18: `InsertFIFOQueue`$(\hat{v}_k)$
19: $w_{start} \leftarrow \arg\min_{w\in W_{\text{start}}} \hat{v}_j$
20: $\zeta \leftarrow \zeta_{w_{start}}$;
21: **return** ζ

B. High Level Algorithms For Hybrid And Iterative Hybrid Path-Based Sensor Information Planning (Problems 4 and 5)

We now present algorithms that solve the problem in which we seek to gather information about both hazards (phenomenon one) using a path-based sensor and information about targets (phenomenon two) using a standard sensor.

The outer loop of the algorithm that solves the iterative version of this problem appears in Algorithm 5. The difference between Algorithm 5 and Algorithm 1 is the dependence of the path on both target belief $\mathbf{X}$ and hazard belief $\mathbf{Z}$ (line 2), and the addition of the update to the target distribution that occurs if and only if the robot survives (line 5), where $\mathbf{Y}_{\zeta}$ is the sequence of sensor observations about targets that the robot gatherd along its path using a typical sensor.

`calculatePath`$(\mathbf{X}, \mathbf{Z})$, which solves Problem 4, appears in Algorithm 6. Algorithm 6 is very similar to Algorithm 2 except that it also considers target beliefs $\mathbf{X}$. This involves calculating the expected information gained about the target distribution along each candidate subpath (line 9), and then combining this with the expected information gained about hazards (line 12) using the user defined combination weights (c_X and c_Z, respectively). The integral that appears on line 9 is deceptively simple in appearance. In Section A of the appendix we discuss how it can be implemented efficiently using dynamic programming.

C. Optimal But Computationally Complex Algorithms

We now describe optimal algorithms (Algorithms 7 and 8) that use the same basic framework as Algorithms 2 and 6, respectively. The main differences between the optimal algorithms and their counterparts are the following: instead of using the graph $E_{\mathbf{S}\times\mathbb{T}}$, which contains $|V_{\mathbf{S}\times\mathbb{T}\times\mathbb{H}}| = \ell|V_{\mathbf{S}}|$ nodes, we must use a different graph $G_{\mathbf{S}\times\mathbb{T}\times\mathbb{H}}$ that contains all possible valid histories of reaching each goal node from each start node. In general, $G_{\mathbf{S}\times\mathbb{T}\times\mathbb{H}}$ contains $|E_{\mathbf{S}\times\mathbb{T}\times\mathbb{H}}| = \Omega(c^{\ell}|V_{\mathbf{S}}|)$ nodes, where c is the number of neighbors of each node, i.e., the branching factor.

Algorithm 7 `calculateOptimalPath`$(\mathbf{Z})$
Input: Hazard beliefs $\mathbf{Z}$
Output: Path ζ

1: **for all** $\hat{v}_i \in V_{\mathbf{S}\times\mathbb{T}}$ **do**
2: $h_{\hat{v}_i} = -\infty$
3: **for all** uplink points $w \in W_{\text{goal}}$ **do**
4: $\zeta_w \leftarrow \emptyset$
5: `InsertFIFOQueue`(w)
6: **while** $\hat{v}_i \leftarrow$ `PopFIFOQueue` **do**
7: **for all** $(\hat{v}_i, \hat{v}_j) \in E_{\mathbf{S}\times\mathbb{T}\times\mathbb{H}}$ **do**
8: $\zeta \leftarrow (\hat{v}_i, \hat{v}_j) + \zeta_{\hat{v}_j}$
9: // Note: $E_{\mathbf{S}\times\mathbb{T}\times\mathbb{H}}$ contains all possible histories of length ℓ
10: $(\mathbf{Z}_{live}, p^{\text{alive}}_{\zeta}) \leftarrow$ `KilledOnPathUpdate`$(\mathbf{Z}, \zeta)$
11: $\mathbf{Z}_{killed} \leftarrow$ `BayesianCellUpdates`$(\mathbf{Z}, [0, \ldots, 0])$
12: $\hat{h}_{this} \leftarrow p^{\text{alive}}_{\zeta} H(\mathbf{Z}_{live}) + (1 - p^{\text{alive}}_{\zeta})H(\mathbf{Z}_{killed})$
13: **if** $\hat{h}_{this} > h_{\hat{v}_i}$ **then**
14: $\zeta_{\hat{v}_i} \leftarrow \zeta$
15: $h_{\hat{v}_i} \leftarrow h_{this}$
16: **for all** $(\hat{v}_k, \hat{v}_i) \in E_{\mathbf{S}\times\mathbb{T}}$ **do**
17: **if not** `InQueue`$(\hat{v}_k)$ **then**
18: `InsertFIFOQueue`$(\hat{v}_k)$
19: $w_{start} \leftarrow \arg\min_{w\in W_{\text{start}}} \hat{v}_j$
20: **return** $\zeta_{w_{start}}$

Algorithm 8 `calculateOptimalPath`$(\mathbf{X}, \mathbf{Z})$
Input: Target beliefs $\mathbf{X}$ and hazard beliefs $\mathbf{Z}$
Output: Path ζ

1: **for all** $\hat{v}_i \in V_{\mathbf{S}\times\mathbb{T}}$ **do**
2: $h_{\hat{v}_i} = -\infty$
3: **for all** uplink points $w \in W_{\text{goal}}$ **do**
4: $\zeta_w \leftarrow \emptyset$
5: `InsertFIFOQueue`(w)
6: **while** $\hat{v}_i \leftarrow$ `PopFIFOQueue` **do**
7: **for all** $(\hat{v}_i, \hat{v}_j) \in E_{\mathbf{S}\times\mathbb{T}\times\mathbb{H}}$ **do**
8: // Note: $E_{\mathbf{S}\times\mathbb{T}\times\mathbb{H}}$ contains all possible histories of length ℓ
9: $\zeta \leftarrow (\hat{v}_i, \hat{v}_j) + \zeta_{\hat{v}_j}$
10: $\hat{h}^{\mathcal{X}}_{live} \leftarrow \int_{x\in\mathcal{X}} H(\mathbf{X}_{live})dx$
11: $(\mathbf{Z}_{live}, p^{\text{alive}}_{\zeta}) \leftarrow$ `KilledOnPathUpdate`$(\mathbf{Z}, \zeta)$
12: $\mathbf{Z}_{killed} \leftarrow$ `BayesianCellUpdates`$(\mathbf{Z}, [0, \ldots, 0])$
13: $\hat{h}_{this} \leftarrow c_Z(p^{\text{alive}}_{\zeta} H(\mathbf{Z}_{live}) + (1 - p^{\text{alive}}_{\zeta})H(\mathbf{Z}_{killed})) + c_X p^{\text{alive}}_{\zeta} \hat{h}^{\mathcal{X}}_{live}$
14: **if** $\hat{h}_{this} > h_{\hat{v}_i}$ **then**
15: $\zeta_{\hat{v}_i} \leftarrow \zeta$
16: $h_{\hat{v}_i} \leftarrow h_{this}$
17: **for all** $(\hat{v}_k, \hat{v}_i) \in E_{\mathbf{S}\times\mathbb{T}}$ **do**
18: **if not** `InQueue`$(\hat{v}_k)$ **then**
19: `InsertFIFOQueue`$(\hat{v}_k)$
20: $w_{start} \leftarrow \arg\min_{w\in W_{\text{start}}} \hat{v}_j$
21: **return** $\zeta_{w_{start}}$

Given the submodularity property of information gain, the consideration of all possible histories is required to guarantee that we find the optimal path. The algorithms in this subsection quickly become intractable as path length increases.

VII. Theoretical Runtime and Space Requirements Analysis

We now discuss the runtime and space requirements of our method, the optimal variant, and a faster relaxation of our method. The relaxation method itself is presented in detail in the appendix.

A. Runtime And Space Requirements, Our Method

We require storage for $\mathcal{O}(\ell m)$ nodes in the space-time graph given that we seek a path of length ℓ through a map that contains m map cells. Each cell has some number of neighbors. Let c_{max} be the maximum number of neighbors of any cell. Storage of the space-time graph itself, including nodes and edges, requires $\mathcal{O}(c_{max}\ell m)$ space.

The information theoretic calculation of the next node in a path requires that we consider all possible histories of events local to that cell. Because we assume that events are local to each cell, if we visit a particular cell c_{visits} times, then we must consider the potential outcome of 2^c_{visits} different events (see Figure 8). Because events are local to each cell, we need only update events related to the particular cell that is visited at each new path segment (see Figure 9). The amount of data we track to calculate entropy along any path is $\mathcal{O}(\ell + 2^{c_{mv}})$, where c_{mv} is the greatest number of visits to a particular cell along any particular path.

It is often the case that $c_{mv} \ll \ell$, therefore we parameterize the space requirements using c_{mv}. In our algorithm (which is not optimal), there are at most m partial paths that are tracked at each time step. The total space required is then $\mathcal{O}(c_{max}\ell m + \ell^2 m + \ell m 2^{c_{mv}})$.

The runtime requirements of the algorithm are closely linked to the space requirements. We must calculate the possible predecessors for each subpath, accounting for the possibilities of extending them through each neighbor backward in time. At each of ℓ times there are $\mathcal{O}(mc_{max})$ possible paths, even though only m of them are chosen for extension. Using the dynamic programming scheme described in Algorithm 9 each one of these calculations takes time proportional to the length of the subpath to determine if the new cell has previously been visited, taking $\mathcal{O}(\ell)$ time. This is followed by a determination of the probabilities and entropy values associated with adding another visit, taking $\mathcal{O}(2^{c_{visits}})$ time when the new path segment goes through the cell the c_{visits}-th time.

The total runtime is $\mathcal{O}(\ell^2 m + \ell m c_{max} 2^{c_{mv}})$. In the environments we use in our experiments c_{mv} tends to be a small constant. However, we believe that it is possible to design scenarios that would cause c_{mv} to be large. In particular, c_{mv} will be large whenever $\ell \gg m$.

B. Runtime and Space Requirements, Optimal Method

The main difference between the optimal algorithm and our method is that the optimal algorithm must consider all paths to a particular node in space-time. The optimal method considers all possible $\mathcal{O}(c^\ell_{max})$ histories of length ℓ starting from each of the m different map locations. Moreover, it tracks a number of alternative histories proportional to the maximum number of revisits to the most revisited cell along any particular history. The space requirements are $\mathcal{O}(c_{max}\ell m + \ell \max(\ell m 2^{c_{mv}}, m c^\ell_{max} 2^{c_{mv}}))$. At least one sub-history for each map cell involves simply remaining stationary, in which case $c_{mv} = \ell$, and so the space and time requirements are both $\Omega(m\ell 2^\ell)$.

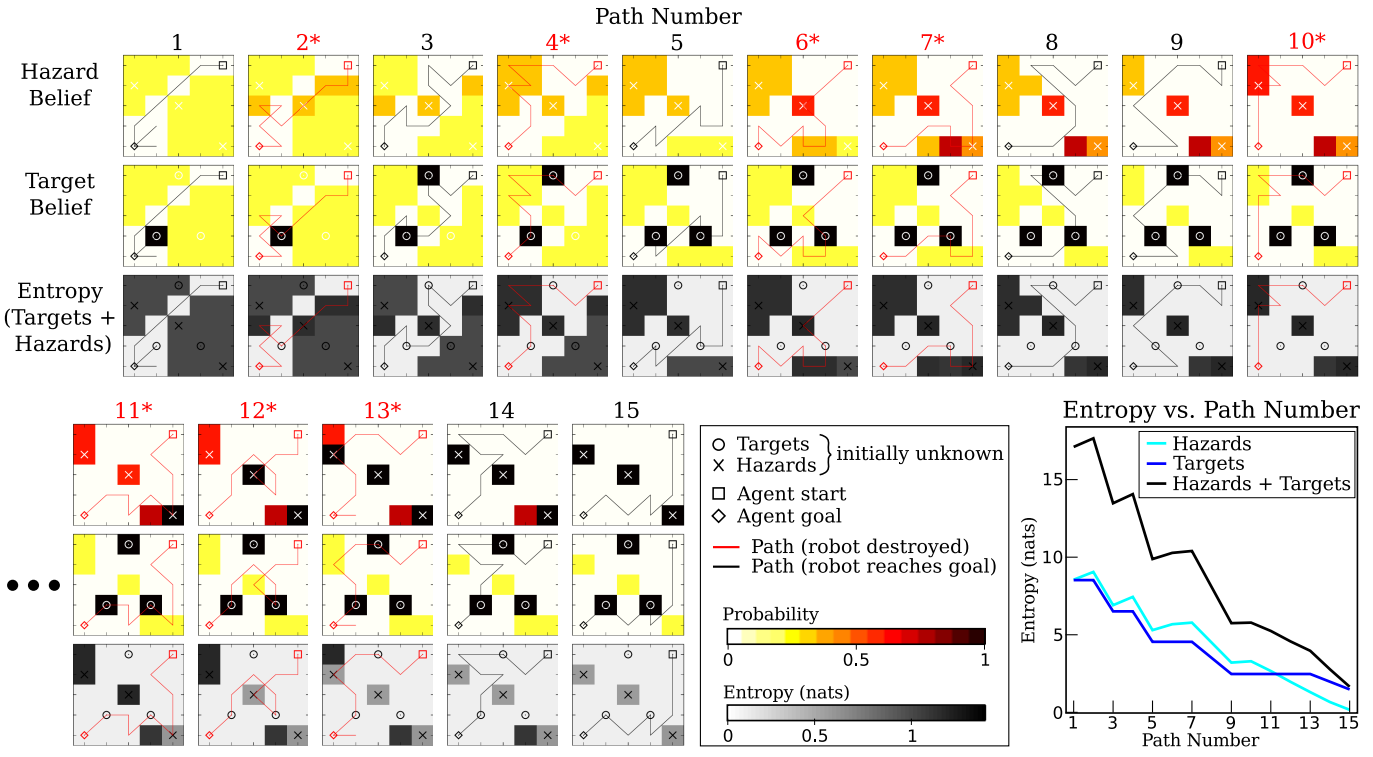

Fig. 3: Fifteen robot paths and the belief and entropy maps that are created from their path-based sensor readings (1-15). Each path is plotted on the map that results from its path-based sensor reading, so each path is planned based on the data in the previous map. Paths along which the robot is destroyed are labeled red and with a '*' after their iteration label. Maps use different color scales for probability (hazards and target belief) and entropy.

C. Runtime and Space Requirements, Heuristic Method

The greedy heuristic ignores repeated visits to the same cell. This allows us to forgo the $\mathcal{O}(2^{c_{mv}})$ storage and calculation of the different outcomes over all possibly revisit histories to any particular node. Space requirements are $\mathcal{O}(c_{max}\ell m+\ell^2 m)$ and time requirements are $\mathcal{O}(\ell^2 m + \ell m c_{max})$.

VIII. Experiments And Discussion Of Results

In Sections VIII-A through VIII-E we describe a number of experiments in which our methods have been evaluated in different scenarios and/or compared to other ideas.

A. Toy Example Involving Problem 5

We begin our evaluation by focusing on a simple toy example of Problem 5. This experiment is included mostly for illustrative purposes, and involves paths constrained to 10 moves in a 5×5 environment where false positive and false negative rates are both 0.01. Travel is defined by 9-grid connectivity; forward one time unit in time and up, down, left, right, diagonal, and stationary in space. Figure 3 shows the first 15 paths and how their path-based sensor readings affect the environmental belief maps and resulting entropy. When robots are destroyed there is no change in the entropy related to targets because that information is destroyed; however, we still get information about hazards—and hazard entropy increases or decreases depending on if the robot's destruction was a surprise or reinforces our beliefs, respectively.

B. Runtime Comparison of Information Gathering Methods

We evaluate the runtime of our method, its relaxation, the optimal version, and greedy information surfing on Problem 5 (Ex 3) in a 5×5 grid environment. Travel is defined by 9-grid connectivity. We increase path length until the optimal algorithm becomes intractable. Experimental results (Figure 4) verify the theoretical result that the optimal method quickly becomes intractable versus path length. Our method and its relaxation have nearly identical runtime in this experiment. Both outperform greedy information surfing with respect to the amount of information gathered but take longer to run.

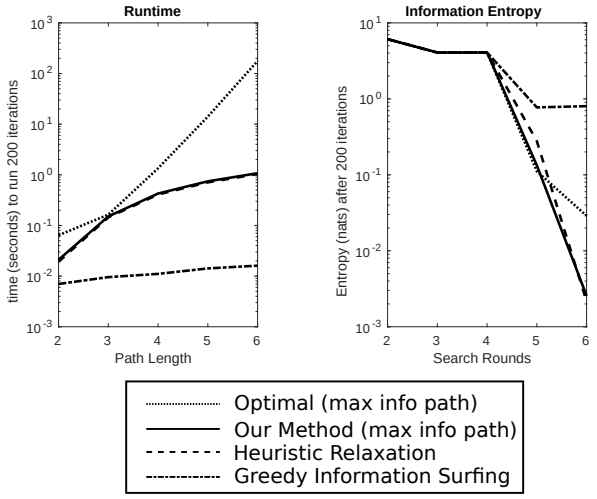

Fig. 4: Left: Runtime comparison. The optimal method becomes intractable after a path length of six. Right: the resulting information entropy. The entropy difference between the first three methods is quite small (note the log scale). The optimal method is only optimal in expectation and performs slightly worse in this trial due to random chance.

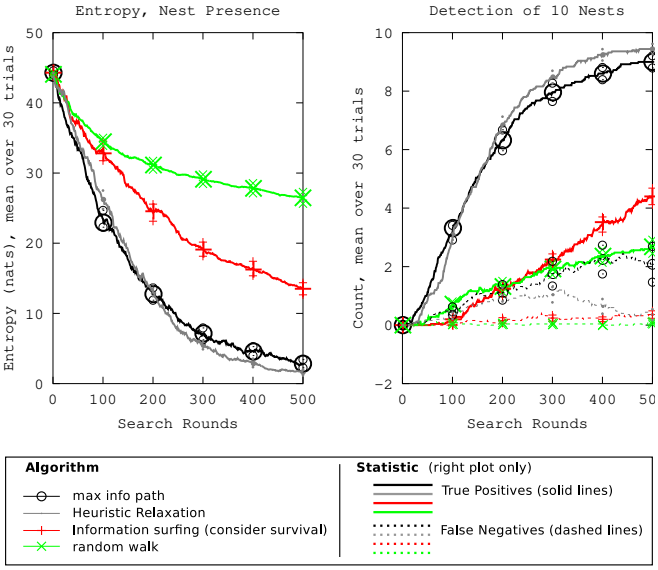

Fig. 5: Entropy values and Statistics on insect nest existence belief statistics (mean over 25 random trials). Color and marker type corresponds to method, while line style corresponds to performance metric (entropy, true-positives based on 95% belief threshold, etc.). Error bars show standard error every 100 paths.

C. Comparison of Methods for Iterative Information Gathering: Searching for an Insect Nest

In this experiment a path-based sensor is used to iteratively collect information about the location of an insect nest in the environment (Problem 2 and Ex 1). We perform Monte Carlo trials in simulation to compare the performance of our method and its heuristic-based relaxation to greedy information surfing as well as a random walk.

The environment is represented by a 15×15 map. Movement is defined by 9-grid connectivity. The agent has fuel for 25 moves. Insect detection false positive rate is $p_{\text{FalsePos}} = .01$ per time step. In expectation a false positive path-sensor reading happens during $1-(1-0.01)^{25} \approx 0.22 = 22\%$ of all forays of length 25. A false positive happens, e.g., when an insect is collected but does not come from the nests we seek. The false negative rate is set to $p_{\text{FalseNeg}} = .7$, meaning that 70% of the time we fly through a cell that contains a nest we will *not* obtain an insect sample. This high false negative rate is chosen to evaluate the usefulness of our methods in the case that observing the phenomena of interest is relatively improbable. In each trial the start and goal are placed uniformly at random. 10 non-start/goal locations are picked uniformly at random (no replacement) and populated with insect nests.

The results from this experiment are presented in Figure 5, and show that both our method and its relaxation outperform

greedy information surfing and a random walk.

D. The Effects of False Negatives and False Positives

In this experiment a path-based sensor is used to iteratively collect information about the location of an insect nest in the environment (Problem 2 and Ex 1). We perform Monte Carlo trials in simulation to compare the performance of our method and its heuristic-based relaxation to greedy information surfing as well as a random walk.

The environment is represented by a 10×10 grid map. Movement is defined by 9-grid of connectivity. The agent has fuel for 10 moves. In one set of trials we hold the false negative rate constant at 0.1 and vary false positive rate per cell traversal across the range $0.001, 0.003, 0.01, 0.033, 0.1, 0.333$. In a second set of trials we hold the false positive rate constant at 0.1 and vary false negative rate per phenomenon containing cell traversal across the range $0.01, 0.2, 0.4, 0.6, 0.8$. Results appear in Figure 6.

E. Comparison of Methods for Iterative Information Gathering: Searching for Targets in a Hazardous Environment

We test our algorithm using three different objective functions: weighting information from targets and hazards equally $c_X, c_Z = \{1, 1\}$; gathering only target information $c_X, c_Z = \{1, 0\}$; and gathering only hazard information $c_X, c_Z = \{0, 1\}$. We compare our method to three other ideas: (1) 1-step look ahead information surfing[2]; (2) a Markov random walk; and (3) planning paths to gather target information while ignoring hazards altogether by not accounting for the probability of being destroyed when evaluating the expected information gain and using a $c_X, c_Z = \{1, 0\}$ objective. In all methods agent movement is only allowed in directions from which the agent can still reach the goal given its fuel reserves.

We evaluate the performance of all seven methods/variants across six hazard lethality rates: 0.01, 0.2, 0.4, 0.6, 0.8, and 0.99 in 15×15 maps with fuel for 25 moves. Two subsets of trials are run that involve different numbers of targets and hazards. In both sets statistics are gathered by testing each combination of method and hazard lethality rate across 30 Monte Carlo trials. The same cell may contain a hazard and a target. True and false positives are calculated based on a 95% belief: if target existence belief is ≥ 0.95 in a cell, then we declare that there is a target in that cell, likewise for hazards.

- Experiment set E.1. Start and goal locations are placed uniformly at random per each of the 30 scenarios. Hazards: 10 non-start/goal locations picked uniformly at random (no replacement). Targets: 20 non-start/goal locations picked uniformly at random (no replacement). Results appear in Figure 7.
- Experiment set E.2. Start and goal locations are selected once, uniformly at random, and then reused across 30 scenarios. Hazards: 4 non-start/goal locations picked uniformly at random (no replacement). Targets: 6 non-start/goal locations picked uniformly at random (no replacement). Results appear in Figure 12 in the appendix.

[2]In "information surfing" the path is greedily computed—the path is initially unknown when that agent leaves the start and then the path is computed on-the-fly. Hazard existence belief is tracked and used to determine the expected information that will be gained about targets. In practice, the destruction of an agent eliminates direct knowledge of the path taken by the agent. While it may be possible to integrate over all possible paths the agent could have taken to obtain a valid update, this computation is at least as hard as the algorithm we present for planning the optimal path. Instead, for the purposes of comparison we choose to be overly generous to "information surfing" and (unrealistically) assume that if the agent is destroyed, then we still know the path that it would have taken to the goal had it not been destroyed. Using this path to refine hazard beliefs (by calculating the relative likelihood the agent was destroyed on each segment) provides a performance bound such that the results for "information surfing" are better than what is expected in practice.

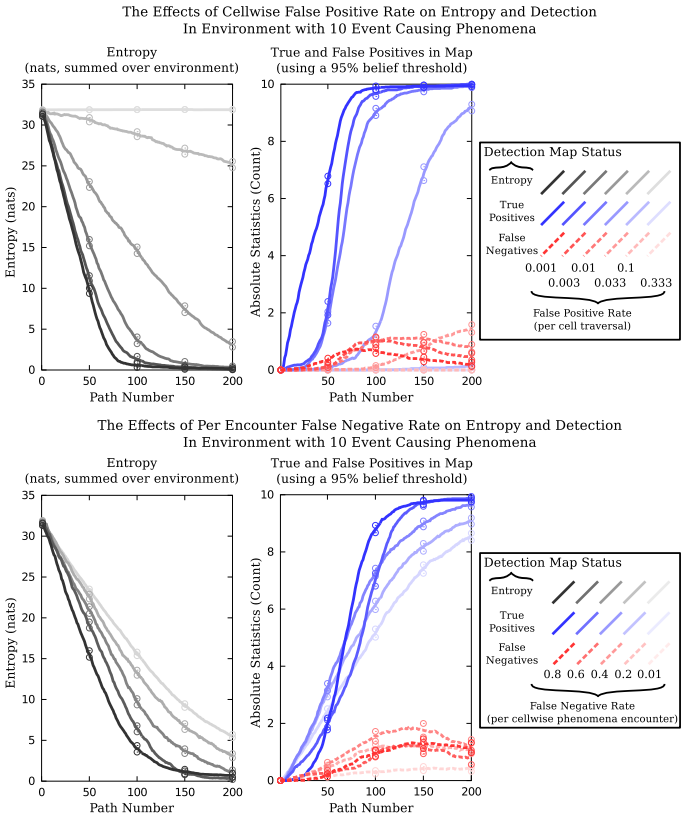

Fig. 6: Performance of our method in response to different false positive rates (top) and false negative rates (bottom). Different rates are drawn with different color saturation. In both cases the other value (false negative or positive, respectively) is held constant at 0.1. Entropy (Left) and true/false positives (Right) of cells identified as containing event causing phenomena based on 95% belief threshold. Error bars show standard error every 50 paths.

We find the relative performance of the various methods to be stable across different hazard lethality rates, and across both sets of trials. Increasing hazard lethality makes hazard detection easier and target detection harder for all methods.

Our path-sensor path planning method with $c_X, c_Z = \{1, 1\}$ objective consistently outperforms the other methods. The $c_X, c_Z = \{0, 1\}$ objective initially has less accurate beliefs regarding *target* existence before improving during later path planning iterations. This makes sense given that this objective explicitly ignores ignores target information.

Not considering the existence of hazards during target search in a hazardous environment yields the worst performance of all methods—even worse than a random walk. This happens because target beliefs remain unchanged in the

event that agents are destroyed—if hazards are ignored then subsequent agents will continue to attempt the same dangerous path until it is successfully completed. We also note that greedy information outperforms a random walk.

Our algorithms work as intended. Planning a path to maximize the expected information gain improves the performance of path-based sensors. Because this is the first presentation of a path-based sensor, the methods to which we compare have been repurposed from different but related problem domains. Thus, the lackluster performance of the other methods with respect to our problem domain—for which our method was specifically designed—should not be taken as a general reflection on the quality of the other methods. The other methods are expected to work well in the problem domains for which they were originally designed.

IX. Conclusion

We have introduced the notion of a path-based sensor and derived its recursive Bayesian update for a discrete time model. We have also presented a number of algorithms that leverage these ideas for Shannon information theoretic path planning.

The recursive Bayesian update accounts for both false positives and false negatives and can be used *regardless* of how the path has been calculated. This means that the path-based sensor Bayesian update can be used when a path-based sensor piggybacks on an unrelated mission—even if we have no control over the path(s) the robot(s) must follow on these missions. Our experiments show that even a path-based sensor that follows a random walk provides useful information for the path-based sensor belief update. That said, designing a path to maximize the information gathered by the path-based sensor improves performance.

Tracking hazards in a dangerous environment enables us to gather more information about a separate phenomenon of interest, especially when robot survival is a prerequesite for returning information about the separate phenomenon of interest. Risky paths are attempted only if the expected information gain is sufficient to justify the risk. Our algorithms can be be modified to handle the case of malfunction in environments *without* hazards by prepopulating the hazard belief map cells with prior values of zero, and then running them as normal. This may be useful in cases where robot malfunction causes the loss of all target information that has been gathering during a path traversal.

Acknowledgment

This work was performed at the U.S. Naval Research Laboratory (NRL) and the University of Maryland (UMD) and funded by Office of Naval Research (ONR) under the efforts “Autonomous Multi-Agent Search and Rescue in Unpredictable Contested Environments” and “Autonomous Path Planning for Information Gathering In Lethally Hostile Environments with Severely Limited Communication”. The views, positions and conclusions expressed herein reflect only the authors’ opinions and expressly do not reflect those of ONR, nor those of NRL and/or UMD.

Appendix
Algorithm Optimizations

In this section we discuss optimizations that enable our algorithms to run quickly in practice. We also describe a relaxation that leads to significant speedup in practice.

Much of the computational complexity of solving Problems 4 and 5 is due to the path entropy integral that pertains to the target beliefs—equation that appears on Line 8 in Algorithm 6:

$$\hat{h}^{\mathcal{X}}_{live} \leftarrow \int_{x\in\mathcal{X}} H(\mathbf{X}_{live})dx$$

This equation is a major bottleneck of solving Problems 4 and 5—as well as any problem in which a typical sensor is used to gather information at multiple points along a path (in particular, in standard non-path-based sensor formulations).

Even if we assume the simplest possible cell-wise interaction, i.e., that targets and hazards only affect sensor readings and robot survivability within their own cells, then repeatedly solving this equation still takes a significant amount of time—especially if a naive approach is used

In Section A we discuss a dynamic programming solution that speeds this calculation. In Section B we describe how similar ideas can be used for the other quantities that require tracking. In Section C we show how a lower bound on the required value can be calculated even more quickly by using a modification to this dynamic programming solution.

It is important to note that the computational efficiency of these methods leverages an assumption that the neighborhood in which targets or hazards cause observable effects is much smaller than the size of the entire environment.

A. Dynamic Programming Calculation of $\int_{x\in\mathcal{X}} H(\mathbf{X}_{live})dx$

The overall structure of Algorithm 6 requires us to calculate a set of all subpaths of length k before extending these to length $k+1$. Assume that we have two subpaths, one of length k and the other of length $k+1$, such that the first and the second are identical at their first k segments. *If* mutual information were *not* submodular—e.g., in a “pretend” universe—then the additional expected mutual information gained along the longer path would simply be the mutual information gained in the final segment of the longer path.

This “pretend” ideal case is not what happens, in general. However, if we can assume that a particular hazard or targets *only* affect the robot and its sensor when the robot is located in (or near) the same cell as that hazard or target, then the ideal pretend case only breaks down when the robot visits a particular cell (or goes near to that cell) more than once during the same path.

Given our assumptions, the expected mutual information gained from all readings relevant to a particular cell depends *only* on the readings that have been made which are applicable to that cell (as well as their relative order in scenarios where effects go beyond a single cell but change as a function of distance), and is independent of the particular path segment(s) index at which the readings were taken.

Thus, use a parallel data structure to track the number of observations that are not mutually independent and, in

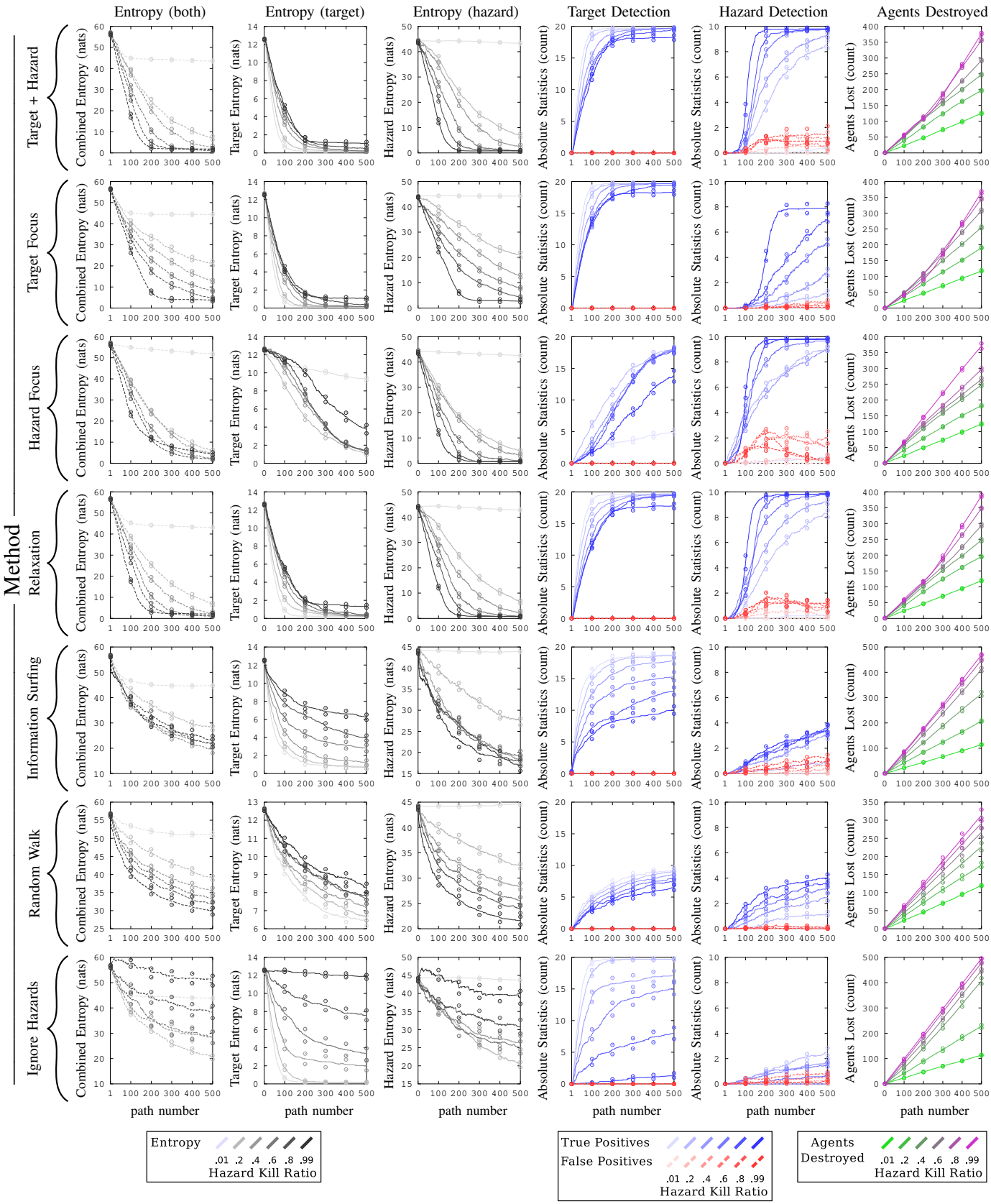

Fig. 7: Performance of various methods across six different hazard kill ratios in a 15 by 15 map with 10 hazards and 20 targets. Different kill ratios have different color saturation values. Row corresponds to method and the top three rows show our method with different objective weights for information about targets versus hazards. Columns correspond to different performance metrics. Performance metrics are also differentiated by color hue value. Statistics include path-based sensor true positives and false positives, with positive detection defined by a $\geq .95$ likelihood value in the map. Each datapoint represents the mean over 30 random trials. Error bars show standard error every 100 paths.

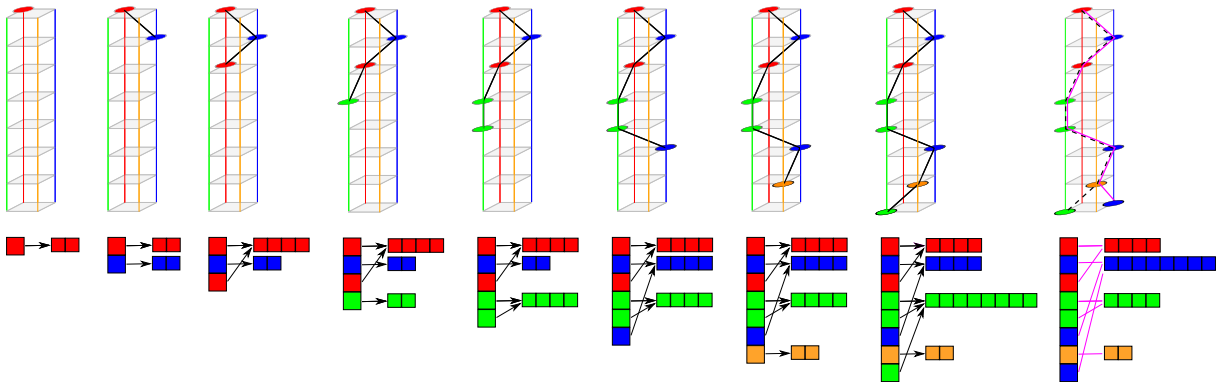

Fig. 8: Visualization of the local histories (different possible sets of outcomes given multiple visits to the same cell) that need to be tracked to calculate the information that is gained along the entire path. Figure 9 shows how this can be done more efficiently. i repeated visits to a particular cell require storing the information that will be gathered in each of the 2^{i-1} different possible observation histories local to that cell. Note that this depiction assumes that target sensors are only influenced by targets when they are in that target's cell.

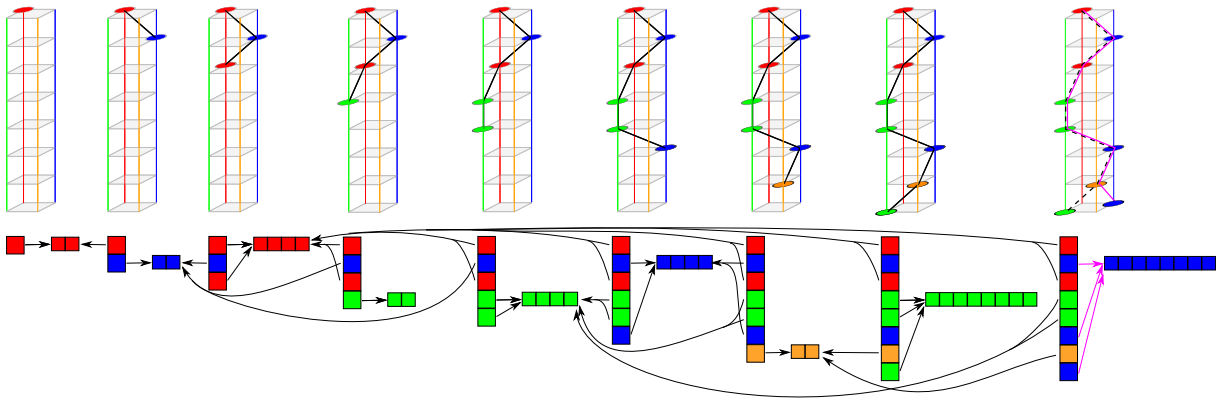

Fig. 9: Visualization of the local histories (different possible sets of outcomes given multiple visits to the same cell) that need to be tracked to calculate the information that is gained along the entire path. This improves the idea presented in Figure 8 because paths only need to recalculate a new (larger) set of histories for each new cell that is added to a path. Note that this depiction assumes that target sensors are only influenced by targets when they are in that target's cell.

scenarios where effects extend beyond a single cell, their order and distance. If, as in our experiments, hazard and target effects are limited to a single cell, then the calculation can be further simplified by realizing that the relative order of visits to a particular cell does not affect the final expected information gained about that cell. All observations are essentially identical, except for the amount of new information they provide.

To calculate the expected information gained along a path, we must track all possible local history of the visits to that cell (see Figure 9). The size of the data will grow exponentially with respect to the number of visits i made to a particular cell, since (for a binary sensor) there are 2^{i-1} possible outcome histories that must be tracked. Thus, data structure size is upper bounded by a function of the number of revisits made to the most revisited cell.

We find that the number of revisits to a particular cell is often small, in practice. However, it is possible to design arbitrarily bad cases by carefully constructing a prior belief that forces a particular cell to have much more information available than the others (see the the next section of this appendix for more details).

Efficiency is gained by recalculating a new (larger) set of $i+1$ histories for the new cell only when a new edge takes the path through a previously visited cell (see Figure 9). This is convenient for a binary sensor because we can calculate the new set of history outcomes relevant to *that* cell by starting with the existing 2^{i-1} possible local history outcomes given i visits and then calculating the two possible history extensions in which the sensor reads 0 or 1, respectively. This results in $2 * 2^{i-1} = 2^i$ local history outcomes at time $i+1$. This idea is used in the dynamic programming implementation of the integral (Algorithm 9).

We now describe Algorithm 9 in detail. We use a C++ like notation such that fields of the node data structure are accessed with the '.' symbol. Each node stores information relevant to the path that goes from itself to the goal. This is used to extract the best path from any node that we have found so far. $\hat{v}_i.\tilde{\mathbf{X}}[:]$ tracks all 2^{c-1} possible cell values that may result given c repeated visits to the cell associate with node $\hat{v}_i$. Similarly, $\hat{v}_i.\tilde{\mathbf{P}}_{X=x}[:]$ tracks the relative probabilities of each of the 2^{c-1} histories occurring.

Let $\zeta[k]$ denote the k-th segment of the path. We start by determining if a new path segment revisits any cell that was previously visited along the path (lines 1-6). If the cell was not previously visited along the path, then we simply populate

Algorithm 9 $integral(\hat{v}_i, \hat{v}_j, \zeta_{\hat{v}_j})$

1: $k \leftarrow -1$
2: **for** $\hat{k} \leftarrow 1, \ldots, |\zeta_{\hat{v}_j}|$ **do**
3: **if** $mapCell(\zeta_{\hat{v}_j}[\hat{k}]) = mapCell(\hat{v}_i)$ **then**
4: $u \leftarrow \zeta_{\hat{v}_j}[\hat{k}]$
5: $k \leftarrow \hat{k}$
6: **break**
7: **if** $k = -1$ **then** // This IS the first visit to cell $\hat{v}_i$ along ζ
8: $k \leftarrow |\zeta_{\hat{v}_j}| + 1$
9: $n_{max} \leftarrow 2$
10: $\hat{v}_i.\tilde{\mathbf{X}}[1:2] \leftarrow [\,\mathbb{P}\left(X_i \,|\, \mathbf{X}[i], 0\right), \mathbb{P}\left(X_i \,|\, \mathbf{X}[i], 1\right)\,]$
11: $\hat{v}_i.\tilde{\mathbf{P}}_{X=x}[1:2] \leftarrow [\,(1.0 - \mathbf{X}[i]), \mathbf{X}[i]\,]$
12: **else** // This is NOT the first visit to cell $\hat{v}_i$ along ζ
13: $n_{max} = |\hat{v}_j.\tilde{\mathbf{P}}_{X=x}[:]|$
14: $\hat{v}_i.\tilde{\mathbf{X}}[1:2n_{max}] \leftarrow \mathbf{0}_{1:2n_{max}}$
15: $\hat{v}_i.\tilde{\mathbf{P}}_{X=x}[1:2n_{max}] \leftarrow \mathbf{0}_{1:2n_{max}}$
16: **for** $n \leftarrow 1 : n_{max}$ **do**
17: $\hat{v}_i.\tilde{\mathbf{X}}[n] \leftarrow \mathbb{P}\left(X_i \,|\, \hat{v}_j.\tilde{\mathbf{X}}[n], 0\right)$
18: $\hat{v}_i.\tilde{\mathbf{P}}_{X=x}[n] \leftarrow \hat{v}_j.\tilde{\mathbf{X}}[n] * (1.0 - \mathbf{X}[i])$
19: $\hat{v}_i.\tilde{\mathbf{X}}[n] \leftarrow \mathbb{P}\left(X_i \,|\, \hat{v}_j.\tilde{\mathbf{X}}[n], 1\right)$
20: $\hat{v}_i.\tilde{\mathbf{P}}_{X=x}[n_{max} + n] \leftarrow \hat{v}_j.\tilde{\mathbf{X}}[n] * \mathbf{X}[i]$
21: $\tilde{\mathbf{H}} \leftarrow \sum_{n=1}^{n_{max}} \hat{v}_i.\tilde{\mathbf{P}}_{X=x}[n] * H(\hat{v}_i.\tilde{\mathbf{X}}[n])$
22: $\hat{v}_i.\tilde{I}[k] \leftarrow H(\mathbf{X}[i]) - \tilde{\mathbf{H}}$
23: **if** $|\zeta_{\hat{v}_j}| = 0$ **then** // The path ζ is equivalent to $\hat{v}_i$
24: $\hat{v}_i.\tilde{I}_{total} \leftarrow \hat{v}_i.\tilde{I}[k]$
25: **else if** $k = -1$ **then** // This IS the first visit to cell $\hat{v}_i$ along ζ
26: $\hat{v}_i.\tilde{I}_{total} \leftarrow \hat{v}_j.\tilde{I}_{total} + \hat{v}_i.\tilde{I}[k]$
27: **else** // This is NOT the first visit to cell $\hat{v}_i$ along ζ
28: $\hat{v}_i.\tilde{I}_{total} \leftarrow \hat{v}_j.\tilde{I}_{total} + \hat{v}_i.\tilde{I}[k] - u.\tilde{I}[k]$
29: **return** $\hat{v}_i.\tilde{I}_{total}$

$\hat{v}_i.\tilde{\mathbf{X}}[1:2]$ and $\hat{v}_i.\tilde{\mathbf{P}}_{X=x}[1:2]$ with both of the possible history outcomes and their relative probabilities, respectively (lines 7 to 11).

On the other hand, if this is not the first visit to this cell, then the k-th node of the path also visited this cell and, as the first time this node's cell was added to this path, it holds the previous set of history outcomes for *all* previous (local) visits to the map cell. For each set of history outcomes we extend the history by one (doubling the size of the history outcomes) by considering all possibilities—either we get a sensor reading at the new node or we do not (lines 12-20). We update the total expected information gained along the path by determining how much new information the new visit to the final node's cell will add (lines 20-29).

B. Dynamic Programming Calculations of Updates

The discussion in Section A is more applicable to the use of a standard sensor (target case) than the case of a path-based sensor (hazard case). In the algorithms presented in Section VI we have simplified the path planning for information about hazard computation in a different way—by ignoring how events that happen on-the-fly affect our hazard beliefs. We do not account for the fact that *if* the robot moved through a particular cell without being destroyed early in the path *then* this should correlate with an increased chance of moving safely through the same cell again later in the path, since it provides additional evidence that the cell is hazard free. Instead, we only update beliefs once the robot returns to base or is presumed to be destroyed.

The calculation of the Bayesian updates that occur if the robot survives to a particular node in the path can be calculated iteratively, as was done for the information integral in Section A. We again leverage the assumption that events are local to cells, and so Bayesian updates only affect the cells that the path visits. Adding a node to the path only requires recalculating the Bayesian update in the single cell associated with that node.

C. Heuristic Calculation of Lower Bound on $\int_{x \in \mathcal{X}} H(\mathbf{X}_{live}) dx$

The dynamic programming approach described above suggests a simple lower bound that can be computed in time proportional to path length and branching factor: simply assume that additional visits to a particular cell yield no new information. This can be seen by comparing Figures 9 and 10.

Example Where Our Method Is Suboptimal

In this appendix we present a simple example demonstrating a situation in which our method produces the suboptimal path. This is depicted in Figure 11. Consider a 1-D environment with five nodes across the space at each time step. These imply the existence of five nodes at each time step of a space-time graph. The space-time plot of the search graph found by our method is depicted (Left). The optimal path (Right), spends an equal amount of time sampling at high information locations a and e, transferring sides halfway through at $\ell/2$. The inner three (orange) locations in column a and e have belief values that are very close to either 1 or 0, and so there is little information to be gained by sampling them. The outer (blue) locations in columns b, c, and d have belief values ≈ 0.5 and so there is maximum information to be gained by sampling them. Information to be gained at a is an infinitesimal amount larger than information to be gained at e, which is why the nodes at column c opt to choose nodes in column b as their parent instead of those in column d. As long as the marginal information to be gained with another visit to column a (resp. e) is greater than the marginal information to be gained by visiting the nodes in column c, then nodes in column a and b (resp. e and d) will choose nodes in column a (resp. e) as their parents. This is suboptimal after $\ell/2$, but continues to happen because of the locally optimal—but globally sub-optimal—tendency to avoid the very low-information column c. This ceases to be the case at time 2 in this "worst case" example. If the path were longer, than the path would eventually switch to sampling the other side, since it contains much information to be gained.

Additional Experiment Figures

Figure 12 presents experimental data from experiment set E.2, which involves Problem 5 (scenario Ex 3) and is described in Section VIII-E.

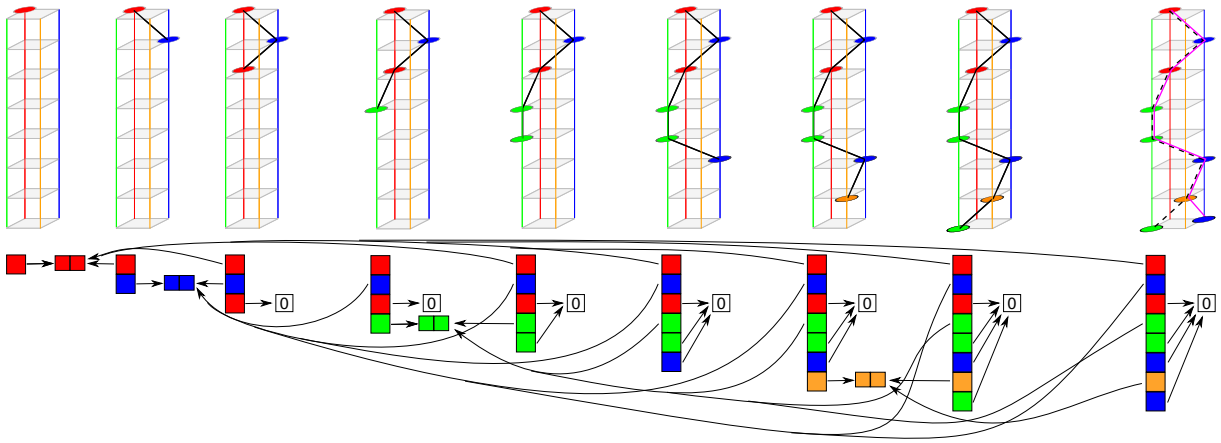

Fig. 10: Much less storage and computation is needed (compared to Figure 9) if we are willing to ignore the effects of repeated visits to the same cell (find a suboptimal solution). In the extreme case depicted here, we assume that no additional information is provided by repeated visits to the same cell. Note that this depiction assumes that target sensors are only influenced by targets when they are in that target's cell.

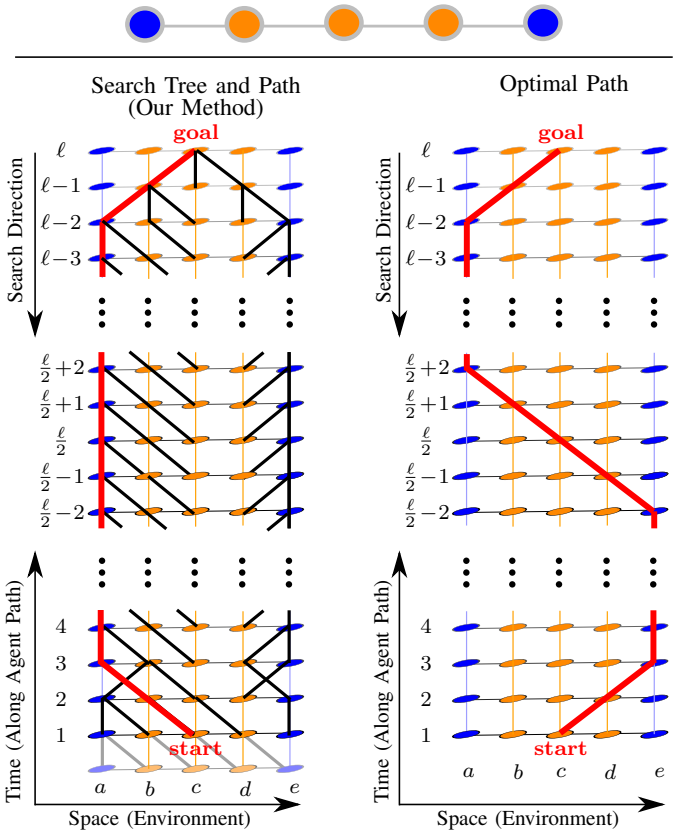

Fig. 11: Example where our method produces the suboptimal path. The 1-D environment contains five nodes. The space-time plot of the search graph found by our method is depicted (Left). The optimal path (Right), spends an equal amount of time sampling both of the high information nodes a and e, transferring sides halfway through at $\ell/2$. The inner three (orange) nodes (in column a and e) have belief values that are very close to either 1 or 0, and so there is little information to be gained by sampling them. The outer (blue) nodes (in columns b, c, and d) have belief values ≈ 0.5 and so there is maximum information to be gained by sampling them.

REFERENCES

[1] T. M. Cover and J. A. Thomas, "Elements of information theory 2nd edition," *Willey Interscience: NJ*, 2006.

[2] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE mobile computing and communications review*, vol. 5, no. 1, pp. 3–55, 2001.

[3] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Autonomous Robots*, vol. 31, no. 4, pp. 299–316, 2011.

[4] C. Robin and S. Lacroix, "Taxonomy on multi-robot target detection and tracking," in *Workshop on Multi-Agent Coordination in Robotic Exploration*, 2014.

[5] M. Schwager, P. Dames, D. Rus, and V. Kumar, *A Multi-robot Control Policy for Information Gathering in the Presence of Unknown Hazards*. Cham: Springer International Publishing, 2017, pp. 455–472.

[6] B. J. Julian, M. Angermann, M. Schwager, and D. Rus, "Distributed robotic sensor networks: An information-theoretic approach," *The International Journal of Robotics Research*, vol. 31, no. 10, pp. 1134–1154, 2012.

[7] P. Dames, M. Schwager, V. Kumar, and D. Rus, "A decentralized control policy for adaptive information gathering in hazardous environments," in *IEEE Conference on Decision and Control*, Dec 2012, pp. 2807–2813.

[8] P. M. Dames, M. Schwager, D. Rus, and V. Kumar, "Active magnetic anomaly detection using multiple micro aerial vehicles," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 153–160, 2016.

[9] P. Dames and V. Kumar, "Autonomous localization of an unknown number of targets without data association using teams of mobile sensors," *Automation Science and Engineering, IEEE Transactions on*, vol. 12, no. 3, pp. 850–864, 2015.

[10] G. A. Hollinger, S. Yerramalli, S. Singh, U. Mitra, and G. S. Sukhatme, "Distributed data fusion for multirobot search," *Transactions on Robotics*, vol. 31, no. 1, pp. 55–66, 2015.

[11] M. Stachura and E. Frew, "Communication-aware information-gathering experiments with an unmanned aircraft system," *Journal of Field Robotics*, vol. 34, no. 4, pp. 736–756, 2017.

[12] B. Grocholsky, "Information-theoretic control of multiple sensor platforms," Ph.D. dissertation, University of Sydney, Australia, 2002.

[13] J. Tisdale, Z. Kim, and J. K. Hedrick, "Autonomous UAV path planning and estimation," *Robotics & Automation Magazine, IEEE*, vol. 16, no. 2, pp. 35–42, 2009.

[14] A. Khan, E. Yanmaz, and B. Rinner, "Information exchange and decision making in micro aerial vehicle networks for cooperative search," *Control of Network Systems, IEEE Transactions on*, vol. 2, no. 4, pp. 335–347, 2015.

[15] J. Berger and J. Happe, "Co-evolutionary search path planning under constrained information-sharing for a cooperative unmanned aerial vehicle team," in *Evolutionary Computation, IEEE Congress on*, July 2010, pp. 1–8.

[16] A. Arora, P. M. Furlong, R. Fitch, S. Sukkarieh, and T. Fong, "Multi-modal active perception for information gathering in science missions," *Autonomous Robots*, vol. 43, no. 7, pp. 1827–1853, 2019.

[17] J. Finke, K. M. Passino, and A. Sparks, "Cooperative control via task load balancing for networked uninhabited autonomous vehicles," in *Decision and Control, IEEE Conference on*, vol. 1, Dec 2003, pp. 31–36.

[18] J. P. Hespanha and H. Kizilocak, "Efficient computation of dynamic probabilistic maps," in *Mediterranean Conference on Control and Automation*, 2002.

[19] S. Waharte, N. Trigoni, and S. J. Julier, "Coordinated search with a swarm of UAVs," in *Sensor, Mesh and Ad Hoc Communications and Networks Workshops, IEEE Communications Society Conference on*. IEEE, 2009, pp. 1–3.

[20] F. Bourgault, T. Furukawa, and H. E. Durrant-Whyte, "Coordinated decentralized search for a lost target in a Bayesian world," in *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, vol. 1. IEEE, 2003, pp. 48–53.

[21] T. H. Chung and J. W. Burdick, "Multi-agent probabilistic search in a sequential decision-theoretic framework," in *Robotics and Automation, IEEE International Conference on*, May 2008, pp. 146–151.

[22] ——, "Analysis of search decision making using probabilistic search strategies," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 132–144, Feb 2012.

[23] A. Khan, E. Yanmaz, and B. Rinner, "Information merging in multi-UAV cooperative search," in *Robotics and Automation, IEEE International Conference on*. IEEE, 2014, pp. 3122–3129.

[24] T. Furukawa, F. Bourgault, B. Lavis, and H. F. Durrant-Whyte, "Recursive Bayesian search-and-tracking using coordinated UAVs for lost targets," in *Robotics and Automation, IEEE International Conference on*, May 2006, pp. 2521–2526.

[25] F. M. Delle Fave, Z. Xu, A. Rogers, and N. R. Jennings, "Decentralised coordination of unmanned aerial vehicles for target search using the max-sum algorithm," in *International Conference on Autonomous Agents and Multiagent Systems*, 2010.

[26] G. A. Hollinger, S. Yerramalli, S. Singh, U. Mitra, and G. S. Sukhatme, "Distributed coordination and data fusion for underwater search," in *Robotics and Automation, IEEE International Conference on*, May 2011, pp. 349–355.

[27] R. Mahler, "'statistics 102' for multisource-multitarget detection and tracking," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 7, no. 3, pp. 376–389, 2013.

[28] L. F. Bertuccelli and J. P. How, "Robust UAV search for environments with imprecise probability maps," in *Decision and Control, European Control Conference, IEEE Conference on*, Dec 2005, pp. 5680–5685.

[29] J. Hu, L. Xie, K. Y. Lum, and J. Xu, "Multiagent information fusion and cooperative control in target search," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1223–1235, July 2013.

[30] M. Flint, M. Polycarpou, and E. Fernandez-Gaucherand, "Cooperative control for multiple autonomous UAVs searching for targets," in *IEEE Conference on Decision and Control*, vol. 3, Dec 2002, pp. 2823–2828.

[31] H. Yetkin, C. Lutz, and D. Stilwell, "Utility-based adaptive path planning for subsea search," in *OCEANS'15 MTS/IEEE Washington*. IEEE, 2015, pp. 1–6.

[32] A. Shende, M. J. Bays, and D. J. Stilwell, "Toward a mission value for subsea search with bottom-type variability," in *Oceans, 2012*, Oct 2012, pp. 1–5.

[33] H. Zhang and Y. Vorobeychik, "Submodular optimization with routing constraints," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[34] M. Mirzaei, F. Sharifi, B. W. Gordon, C. A. Rabbath, and Y. M. Zhang, "Cooperative multi-vehicle search and coverage problem in uncertain environments," in *Decision and Control and European Control Conference, IEEE Conference on*, Dec 2011, pp. 4140–4145.

[35] M. Flint, E. Fernandez-Gaucherand, and M. Polycarpou, "Stochastic modeling of a cooperative autonomous UAV search problem," *Military Operations Research Journal*, 2003.

[36] M. Flint, E. Fernández-Gaucherand, and M. Polycarpou, "Cooperative control for UAVs searching risky environments for targets," in *IEEE Conference on Decision and Control*, vol. 4. IEEE, 2003, pp. 3567–3572.

[37] R. W. Beard and T. W. McLain, "Multiple UAV cooperative search under collision avoidance and limited range communication constraints," in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 1, Dec 2003, pp. 25–30.

[38] C. W. Lum, J. Vagners, and R. T. Rysdyk, "Search algorithm for teams of heterogeneous agents with coverage guarantees," *Journal of Aerospace Computing, Information, and Communication*, vol. 7, no. 1, pp. 1–31, 2010.

[39] R. F. Dell, J. N. Eagle, G. H. Alves Martins, and A. G. Santos, "Using multiple searchers in constrained-path, moving-target search problems," *Naval Research Logistics (NRL)*, vol. 43, no. 4, pp. 463–480, 1996.

[40] F. Bourgault, T. Furukawa, and H. F. Durrant-Whyte, "Decentralized Bayesian negotiation for cooperative search," in *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, vol. 3. IEEE, 2004, pp. 2681–2686.

[41] D. G. Rajnarayan and D. Ghose, "Multiple agent team theoretic decision-making for searching unknown environments," in *Decision and Control, IEEE Conference on*, vol. 3, Dec 2003, pp. 2543–2548.

[42] P. B. Sujit and D. Ghose, "Negotiation schemes for multi-agent cooperative search," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 223, no. 6, pp. 791–813, 2009.

[43] ——, "Search using multiple UAVs with flight time constraints," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 2, pp. 491–509, April 2004.

[44] P. Sujit and D. Ghose, "Multiple UAV search using agent based negotiation scheme," in *American Control Conference, Proceedings of the*. IEEE, 2005, pp. 2995–3000.

[45] ——, "Optimal uncertainty reduction search using the k-shortest path algorithm," in *American Control Conference, Proceedings of the*, vol. 4. IEEE, 2003, pp. 3269–3274.

[46] Y. Yang, A. A. Minai, and M. M. Polycarpou, "Analysis of opportunistic method for cooperative search by mobile agents," in *Decision and Control, IEEE Conference on*, vol. 1, Dec 2002, pp. 576–577.

[47] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, Jun. 1989. [Online]. Available: http://dx.doi.org/10.1109/2.30720

[48] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Robotics and Automation, IEEE International Conference on*, vol. 1, 2000, pp. 476–481.

[49] P. B. Sujit and D. Ghose, "Multiple agent search of an unknown environment using game theoretical models," in *American Control Conference, Proceedings of the*, vol. 6, June 2004, pp. 5564–5569.

[50] Q. Yang and S. Yoo, "Optimal UAV path planning: Sensing data acquisition over IoT sensor networks using multi-objective bio-inspired algorithms," *IEEE Access*, vol. 6, pp. 13 671–13 684, 2018.

[51] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and Service Robotics*. Springer, 1998, pp. 203–209.

[52] S. V. Spires and S. Y. Goldsmith, "Exhaustive geographic search with mobile robots along space-filling curves," in *Collective robotics*. Springer, 1998, pp. 1–12.

[53] P. Vincent and I. Rubin, "A framework and analysis for cooperative search using UAV swarms," in *Proceedings of the ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2004, pp. 79–86.

[54] R. Pěnička, M. Saska, C. Reymann, and S. Lacroix, "Reactive dubins traveling salesman problem for replanning of information gathering by UAVs," in *European Conference on Mobile Robots*, 2017, pp. 1–6.

[55] H. Sato and J. O. Royset, "Path optimization for the resource-constrained searcher," *Naval Research Logistics*, vol. 57, no. 5, pp. 422–440, 2010.

[56] Y. Yang, M. M. Polycarpou, and A. A. Minai, "Multi-UAV cooperative search using an opportunistic learning method," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 716–728, 2007.

[57] Y. Lyu, Y. Chen, and D. Balkcom, "k-survivability: Diversity and survival of expendable robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1164–1171, July 2016.

[58] S. Jorgensen, R. H. Chen, M. B. Milam, and M. Pavone, "The matroid team surviving orienteers problem: Constrained routing of heterogeneous teams with risky traversal," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2017, pp. 5622–5629.

[59] Y. Yang, A. A. Minai, and M. M. Polycarpou, "Decentralized cooperative search by networked UAVs in an uncertain environment," in *American Control Conference*, vol. 6. IEEE, 2004, pp. 5558–5563.

[60] M. Otte and D. Sofge, "Path planning for information gathering with lethal hazards and no communication," in *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Meŕida, Mexico, 2018.

[61] M. J. Kuhlman, M. W. Otte, D. Sofge, and S. K. Gupta, "Multipass target search in natural environments," *Sensors*, vol. 17, no. 11, p. 2514, 2017.

[62] M. Otte, M. Kuhlman, and D. Sofge, "Competitive target search with multi-agent teams: symmetric and asymmetric communication constraints," *Autonomous Robots*, pp. 1–24, 2017.

Michael Otte MICHAEL OTTE (M'07) received the B.S. degrees in aeronautical engineering and computer science from Clarkson University, Potsdam, New York, USA, in 2005, and the M.S. and Ph.D. degrees in computer science from the University of Colorado Boulder, Boulder, CO, USA, in 2007 and 2011, respectively. From 2011 to 2014, he was a Postdoctoral Associate at the Massachusetts Institute of Technology. From 2014 to 2015, he was a Visiting Scholar at the U.S. Air Force Research Lab. From 2016 to 2018, he was a National Research Council RAP Postdoctoral Associate at the U.S. Naval Research Lab. He has been with the Department of Aerospace Engineering, at the University of Maryland, College Park, MD, USA, since 2018 and also holds affiliations with the Maryland Robotics Center, and the Department of Computer Science.

He is the author of over 30 articles, and his research interests include autonomous robotics, motion planning, and multi-agent systems.

Dr. Otte is also a member of the American Institute of Aeronautics and Astronautics, and a member of the IEEE Robotics and Automation Society. He has served as Associate Editor for IEEE conferences including ICRA and IROS.

Donald Sofge DONALD SOFGE (M'00) received the B.S. degree in Mathematical Science (Computer Science) with Physics minor in 1986, and the M.S. degree in Electrical Engineering in 1988.

He is a Computer Scientist and Roboticist at the U.S. Naval Research Laboratory with 32 years of experience in Artificial Intelligence and Control Systems R&D. Prior to joining NRL in 2001 he worked at McDonnell Douglas, NeuroDyne Inc. (a company he co-founded), and GreyPilgrim. He served as a Visiting Scientist at the Massachusetts Institute of Technology from 1991-1993. Currently he leads the Distributed Autonomous Systems Group in the Navy Center for Applied Research in Artificial Intelligence where he develops nature-inspired computing solutions to challenging problems in sensing, artificial intelligence, and control of autonomous robotic systems. His current research focuses on control of autonomous teams or swarms of heterogeneous robotic systems. He has served as an advisor and reviewer on autonomous systems to DARPA, ONR, OSD, ARL, NSF, and NASA, as well as US representative on international TTCP and NATO technical panels on autonomous systems. He served as an advisor to the 2015 Defense Science Board Summer Study on Autonomy, the NITRD Robotics and Intelligent Systems IWG, the DoD AI Strategy Working Group, NITRD Machine Learning and Artificial Intelligence Task Force, and the NITRD Intelligent Robotics & Autonomous Systems IWG. He has served as PI/Co-PI on dozens of federally funded R&D programs, and has authored/co-authored approximately 170 peer-reviewed publications on AI, Robotics, Machine Learning and Quantum Computation, including thirty-one journal articles. He has edited eight books, five journal special issues, and several conference proceedings.

He has served as a reviewer for numerous IEEE conferences and has co-organized workshops for ICRA, IROS, AAAI, RSS, and others. He has been an active member of IEEE since 2000, and has been a Senior Member of IEEE since 2019.

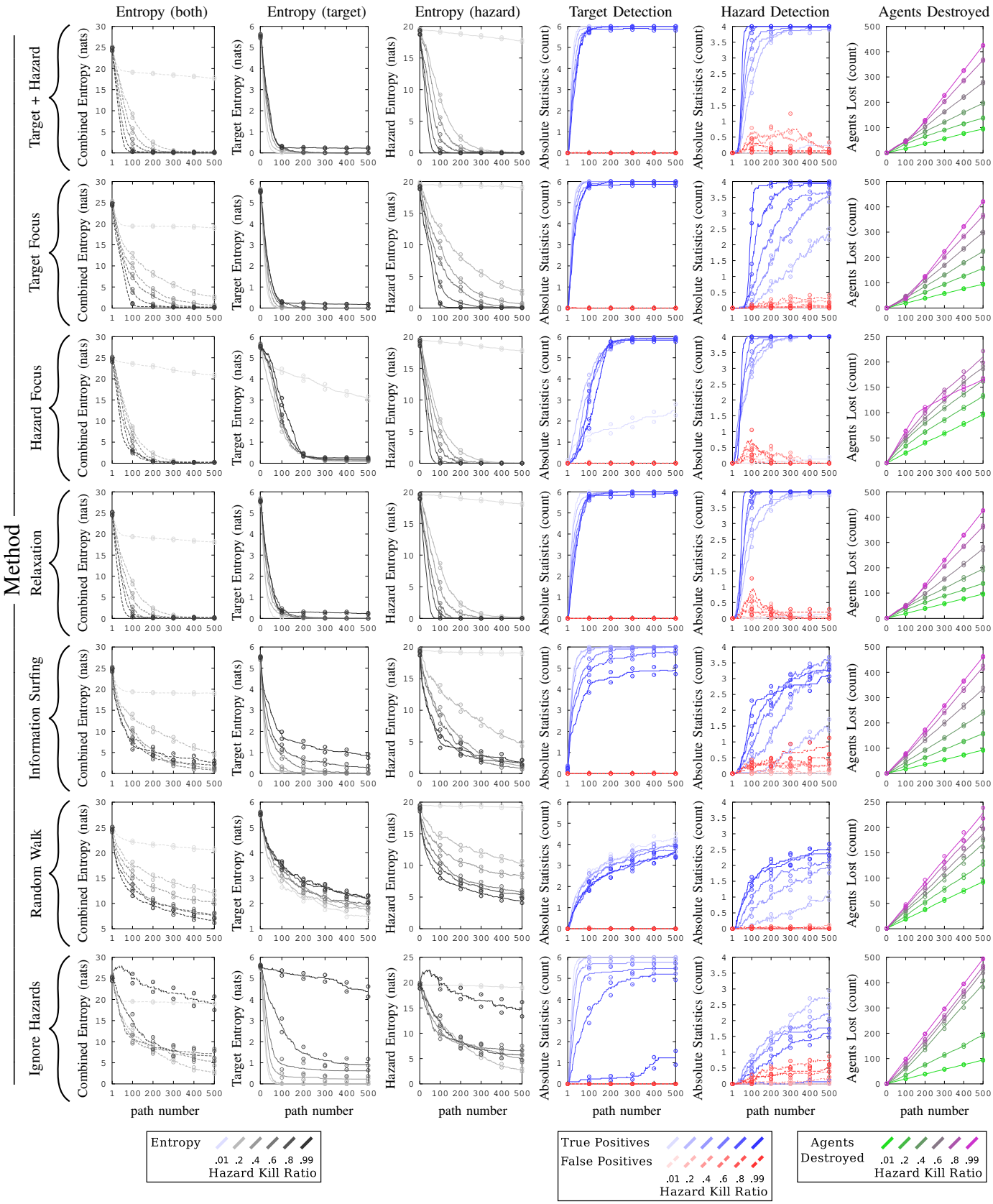

Fig. 12: Performance of various methods across six different hazard kill ratios in a 15 by 15 map with 4 hazards and 6 targets. Different kill ratios have different color saturation values. Row corresponds to method and the top three rows show our method with different objective weights for information about targets versus hazards. Columns correspond to different performance metrics. Performance metrics are also differentiated by color hue value. Statistics include path-based sensor true positives and false positives, with positive detection defined by a $\geq .95$ likelihood value in the map. Each datapoint represents the mean over 30 random trials. Error bars show standard error every 100 paths. Comparing these results to those in Figure 7 we note that the existence of fewer targets and hazards tends to decrease the entropy associated with their existence probability distributions. The detection of all targets and hazards happens more quickly here than in Figure 7.