Surveillance Evasion Through Bayesian Reinforcement Learning

Dongping Qi
Cornell University

David Bindel
Cornell University

Alexander Vladimirsky
Cornell University

Abstract

We consider a task of surveillance-evading path-planning in a continuous setting. An Evader strives to escape from a 2D domain while minimizing the risk of detection (and immediate capture). The probability of detection is path-dependent and determined by the spatially inhomogeneous surveillance intensity, which is fixed but a priori unknown and gradually learned in the multi-episodic setting. We introduce a Bayesian reinforcement learning algorithm that relies on a Gaussian Process regression (to model the surveillance intensity function based on the information from prior episodes), numerical methods for Hamilton-Jacobi PDEs (to plan the best continuous trajectories based on the current model), and Confidence Bounds (to balance the exploration vs exploitation). We use numerical experiments and regret metrics to highlight the significant advantages of our approach compared to traditional graph-based algorithms of reinforcement learning.

1 INTRODUCTION

Path planning is a standard task in robotics, but it becomes much harder if we need to account for stochastic perturbations, adversarial interactions, and incomplete information about the dynamics or the environment. With repeated tasks, Reinforcement Learning (RL) provides a popular framework for optimizing the system performance based on the information accumulated in prior episodes while ensuring the asymptotic convergence to the globally optimal solution as the number of planning episodes grows (Sutton and Barto, 2018). In continuous setting, most applications of RL are focused on learning the process dynamics (Recht, 2019) using frequent or continuous observations of the system state. Our focus here is on a rather different class of problems, where the controlled dynamics are known, but the process termination is a random event, whose probability distribution is not only trajectory-dependent but also a priori unknown.

More specifically, we study the online path planning strategy for an Evader (E), who attempts to escape a region while minimizing the probability of being detected en route. Surveillance intensity imposed by the opponent(s) is assumed to be spatially inhomogeneous, making the choice of E's capture-evading trajectory important. This general setting is motivated by prior work on environmental crime modeling (Cartee and Vladimirsky, 2020) and surveillance avoidance (Gilles and Vladimirsky, 2020; Cartee et al., 2019). However, unlike in those prior papers, here the surveillance intensity is initially unknown to E, who needs to learn it on relevant parts of the domain through multiple planning episodes[1]. Whenever E is spotted, this results in capture and immediately terminates the trajectory. But the evidence obtained about the surveillance intensity on already traced parts of that trajectory can be used to improve the planning in future episodes. Such multi-episodic setting might seem unusual in capture/surveillance avoidance, but it arises naturally in several applied contexts including the environmental crime modeling. E.g., in many parts of Brazil, illegal forrest loggers are primarily subsistence farmers in need of firewood for family use (Chen et al., 2021). When apprehended, their punishment is usually wood confiscation plus sometimes a small fine. Repeat offenders are common, and they also share information with each other on paths taken or locations where they were caught in the past. Another example comes from asymmetries in modern warfare, where many types of UAVs become increasingly cheap – particularly compared to effective air defense systems for large geographic areas.

We develop an approach that balances the exploration against exploitation and uses spatial correlations for efficient learning. Our algorithm relies on numerical meth-

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

[1] We note that our setting is also quite different from the classical *Surveillance-Evasion Games* (Dobbie, 1966; Lewin and Olsder, 1979; Takei et al., 2014), in which the surveillance intensity changes dynamically through adversarial motion of the Observer (O), leading to a differential zero-sum game between E and O, who have immediate and full information of the opponent's actions.

ods for solving PDEs (Sethian, 1996), statistical estimates with censored data (Shorack and Wellner, 2009), Gaussian process (GP) regression (Williams and Rasmussen, 2006), and strategic exploration techniques from RL (Kocsis and Szepesvári, 2006; Azar et al., 2017). To simplify the exposition, our method is described and benchmarked here under the assumption that the Evader is *isotropic* (i.e., E can change the direction of motion instantaneously and the available speed depends on its current position, but not on its chosen direction). However, our main ideas are more broadly applicable, and the approach is also suitable for more realistic (anisotropic) agent dynamics.

We start by reviewing the basic problem with *known* random termination (capture) intensity in section 2. Section 3 poses the problem with *unknown* intensity and defines performance metrics for episodic path planning. We follow this with a review of algorithms for strategic exploration on graphs developed for finite horizon Markov Decision Processes and explain why their usefulness is rather limited in our continuous setting. In section 4 we describe a new approach based on Bayesian models of surveillance intensity function and episodic path planning based on the Confidence Bounds. We show that piecewise-continuous models lead to simpler algorithms but are usually outperformed by models based on GP-regression. The advantages of our methods are illustrated on several sample problems in section 5. We conclude by considering possible future extensions in section 6.

2 PATH PLANNING WITH KNOWN INTENSITY

Suppose E starts at $\boldsymbol{x}$ in some compact domain $\Omega \subset \mathbb{R}^2$ and moves with isotropic speed $f(\boldsymbol{x})$. The motion of E is governed by:

$$\boldsymbol{y}'(s) = f\big(\boldsymbol{y}(s)\big)\boldsymbol{a}(s), \quad \boldsymbol{y}(0) = \boldsymbol{x}, \tag{1}$$

where $\boldsymbol{a} : \mathbb{R} \to S^1$ is a measurable control function specifying the direction of motion at every moment.

Define $T_{\boldsymbol{a}} = \min\{s \geq 0 \mid \boldsymbol{y}(s) \in \partial\Omega\}$ as the domain-exit time if E starts from $\boldsymbol{x}$ and uses the control $\boldsymbol{a}(\cdot)$. The location-dependent surveillance intensity is a smooth, positive function $K(\boldsymbol{x})$, which in this section is assumed to be fully known in advance. If E decides to follow a trajectory $\boldsymbol{y}(\cdot)$, the probability of remaining undetected until time t is

$$\mathbb{P}(S \leq t) = 1 - \exp\left(-\int_0^t K(\boldsymbol{y}(s))ds\right), \tag{2}$$

where S is the random time when E is spotted and immediately captured, thus terminating the trajectory. E's goal is to maximize its probability of reaching $\partial\Omega$ or, equivalently, to minimize the cumulative intensity:

$$\mathcal{J}(\boldsymbol{x}, \boldsymbol{a}(\cdot)) = \int_0^{T_{\boldsymbol{a}}} K(\boldsymbol{y}(s))ds. \tag{3}$$

As usual in dynamic programming, the *value function* $u(\boldsymbol{x})$ is defined to encode the result of optimal choices

$$u(\boldsymbol{x}) = \inf_{\boldsymbol{a}(\cdot)} \mathcal{J}(\boldsymbol{x}, \boldsymbol{a}(\cdot)), \tag{4}$$

and can be found as a solution of a Hamilton-Jacobi-Bellman equation (Bardi and Capuzzo-Dolcetta, 2008). Here we focus on isotropic dynamics/intensity; i.e., f and K do not depend on $\boldsymbol{a}$, which further simplifies the PDE to the following Eikonal equation:

$$\begin{aligned} |\nabla u(\boldsymbol{x})|\, f(\boldsymbol{x}) &= K(\boldsymbol{x}); \\ u\,(\boldsymbol{x}) &= 0, \quad \forall \boldsymbol{x} \in \partial\Omega. \end{aligned} \tag{5}$$

This isotropic setting creates a one-to-one correspondence between a control function and a path. From now on, we interchangeably use the terms "determining a control function" and "selecting a path".

In general, (5) often does not have a classical solution, but always has a unique Lipschitz continuous *viscosity solution* (Bardi and Capuzzo-Dolcetta, 2008). Wherever ∇u exists, the optimal $\boldsymbol{a}$ is opposite to the gradient direction ($\boldsymbol{a}_* = -\nabla u/|\nabla u|$). The set on which u is not differentiable has measure zero and is comprised of all starting positions from which the optimal trajectory to $\partial\Omega$ is not unique.

Efficient numerical methods for solving (5) have been extensively studied in the last 25 years. Many of these algorithms take advantage of the *causality* found in upwind finite-difference discretizations: a gridpoint only depends on its smaller adjacent neighbors, making it possible to solve the system of discretized equations non-iteratively. We choose Fast Marching Method(FMM) (Sethian, 1996), which is a Dijkstra-like algorithm that has $O(N \log N)$ computational complexity when solving (5) on a grid with N gridpoints. Once the value function is approximated, an optimal trajectory can be obtained starting from any $\boldsymbol{x} \in \Omega$ by gradient decent in u until reaching $\partial\Omega$.

3 MULTI-EPISODIC PLANNING WITH UNKNOWN $K(\boldsymbol{x})$

With a known $K(\boldsymbol{x})$, E should persist in choosing a fixed optimal path deterministically, even though the outcomes (whether and where E is captured) may be different every time. But what if $K(\boldsymbol{x})$ is a priori unknown and only learned gradually by trying different paths? If E faces a repeated task of surveillance-avoidance en route to $\partial\Omega$, a natural interpretation is to find a path-selection *policy* that optimizes some long-term performance metric. This is a reinforcement learning (RL) problem, with information gradually collected in a sequence of episodes. Clearly, if E selects enough random trajectories that sufficiently cover the whole region, eventually $K(\boldsymbol{x})$ can be approximately recovered. However, this approach is inefficient since E's

goal is to learn $K(\boldsymbol{x})$ only on those parts of Ω that are relevant to reduce the frequency of captures, asymptotically approaching the probability along the truly optimal trajectory, which would be chosen if $K(\boldsymbol{x})$ were known.

To assess the long-term performance of a policy, a frequently considered criterion is *regret*, which is expected excess of cost due to not selecting the optimal control for all episodes. Letting Δ_i be the indicator of whether E is captured during the ith episode, we define the experimentally observed *excess rate of captures*

$$\mathfrak{S}_j = \frac{1}{j}\sum_{i=1}^{j}\left(\Delta_i - W_*\right), \tag{6}$$

where $W_* = 1 - \exp(-u(\boldsymbol{x}_0))$ is the minimum capture probability. If an optimal $\boldsymbol{a}_*(\cdot)$ were used in each episode, the regret $\mathfrak{S}_j$ would converge to 0 as $j \to \infty$.

3.1 Prior work: RL algorithms on graphs

Before delving into the continuous problem, we first present a discrete version on a finite directed graph G and examine the possibility of applying well-known RL algorithms. Suppose E starts from a node $\boldsymbol{v}_0$, moves between adjacent nodes, and tries to avoid capture en route to a set of target nodes Ξ. We assume that a transition along any edge e incurs some capture probability[2] $\Psi_e \in (0,1)$. If $\mathcal{P}$ denotes the set of all paths from v_0 to Ξ, we would prefer to use $p \in \mathcal{P}$ which maximizes the probability of not being captured up to Ξ; i.e., $\max_{p\in\mathcal{P}} \prod_{e\in p}(1-\Psi_e)$ or, equivalently, $\min_{p\in\mathcal{P}} \sum_{e\in p} -\log(1-\Psi_e)$. When all Ψ_e's are known, this becomes a standard shortest path problem, with $C_e = -\log(1-\Psi_e)$ interpreted as edge weights, and the classical Dijkstra's method can solve it efficiently. Alternatively, this can be viewed as a simple Markov Decision Process (MDP) by adding an absorbing "captured state" node $\boldsymbol{v}_c$. In this MDP interpretation, an action corresponds to a choice of the next attempted edge e, a capture event is modeled as a transition to $\boldsymbol{v}_c$, and a unit reward is earned only upon reaching Ξ.

Of course, we are interested in the case where Ψ_es are *not* known in advance, and it would seem natural to address this by any of the RL algorithms developed to maximize the expected return in MDPs with unknown transition functions. This includes the Q-learning (Watkins and Dayan, 1992), temporal difference methods (Sutton, 1988), Thompson sampling (Thompson, 1933), and the techniques based on *upper confidence bounds* (UCB) (Auer, 2002). The latter served as a basis for a popular model-free Upper Confidence bounds on Trees (UCT) algorithm (Kocsis and Szepesvári, 2006), in which the evidence gathered in previous episodes is used to estimate the rewards of all (state, action) pairs, but the exploration of less visited pairs is encouraged by adding a bonus term proportional to each estimate's standard deviation. The same idea is also used in more recent model-based methods (Dann and Brunskill, 2015; Azar et al., 2017), in which prior evidence is used to model the transition probabilities and the value function is computed in each episode based on the current model but with similar bonus terms added to encourage the exploration.

While the above algorithms were originally designed for MDPs with a fixed finite horizon, they can also be adapted to our "exit time" case. For simplicity, suppose that Ω is discretized using a uniform Cartesian grid of nodes V, with each interior node $\boldsymbol{v}$ connected by edges to its 8 closest neighbors – corresponding to 8 actions (directions of motion) available at that node. We will use $\mathcal{E}(\boldsymbol{v})$ to denote all edges available at $\boldsymbol{v}$ and $\mathcal{E}$ for the set of all edges in the graph. Using the MDP interpretation, the process terminates upon reaching $\Xi \subset V$ (which discretizes $\partial\Omega$) or $\boldsymbol{v}_c$ (in case of capture). To implement UCT, we maintain the statistics $N_{\boldsymbol{v}}$ (and N_e) on how many times each state (and each edge – or state-action pair) is visited over multiple episodes, with Q_e encoding the fraction of those visits on which E traversed e but was captured before reaching Ξ. The selection of nodes is summarized in Algorithm 1, with the parameter $\lambda > 0$ regulating the rate of exploration.

Algorithm 1 UCT: model-free planning on a graph

Set $Q_e = 0,\ N_e = 0,\ N_{\boldsymbol{v}} = 0$ for all $\boldsymbol{v}$ and e
while $t = 1 : T$ **do**
  capture = `search`$(\boldsymbol{v}_0)$;
end while

Function `search` :
Input: current node $\boldsymbol{v} \notin \Xi$
Output: capture flag c
$N_{\boldsymbol{v}} = N_{\boldsymbol{v}} + 1$;
$\hat{e} = \arg\min_{e\in\mathcal{E}(\boldsymbol{v})} Q_e - \lambda\sqrt{\log(N_{\boldsymbol{v}})/\max(N_e, 1)}$;
$N_{\hat{e}} = N_{\hat{e}} + 1$;
$\hat{\boldsymbol{v}} =$ `attempt_transition`$(\boldsymbol{v}, \hat{e})$;
if $\hat{\boldsymbol{v}} == \boldsymbol{v}_c$ **then**
  $c = 1$;
else if $\hat{\boldsymbol{v}} \in \Xi$ **then**
  $c = 0$;
else
  $c =$ `search`$(\hat{\boldsymbol{v}})$;
end if
$Q_{\hat{e}} = \left[(N_{\hat{e}} - 1)Q_{\hat{e}} + c\right]/N_{\hat{e}}$;

The model-based version on a graph (which we will denote Alg-D, for "discrete") is implemented on the same grid, but relies on learning Ψ_e for the relevant edges. We maintain statistics N_e (and ϕ_e) on how many times a visit (and capture) happen for each edge e. An estimate of Ψ_e is com-

[2] If the graph is embedded in a continuous domain Ω, this Ψ_e can be computed from (2), provided t is the time needed to traverse that edge e, which is parametrized by a path $\boldsymbol{y}(s)$.

puted as $\tilde{\Psi}_e = \phi_e / N_e$ and the confidence-bound-modified version is

$$\hat{\Psi}_e = \max\left\{\underline{\Psi},\ \tilde{\Psi}_e - \sqrt{\frac{\log(T|\mathcal{E}|/\gamma)}{\max(N_e, 1)}}\right\}, \qquad (7)$$

where $|\mathcal{E}|$ is the total number of edges, and $\underline{\Psi} \geq 0$ is a known lower bound on all Ψ_e, while $\gamma \in (0, 1)$ is a parameter controlling the decay of expected regret[3]. In each episode, we solve a shortest path problem based on edge weights $\hat{C}_e = -\log(1 - \hat{\Psi}_e)$. Then we simulate running through the derived optimal path and update edge statistics N_e and ϕ_e, which are used to change $\hat{C}_e$ in the next episode. The resulting method is summarized in Algorithm 2.

Algorithm 2 Alg-D: model-based planning on a graph

Set $\phi_e = 0,\ N_e = 0$ for all e.
while $t = 1 : T$ **do**
  Update $\hat{\Psi}_e$ according to (7);
  Solve the deterministic shortest path problem with edge costs $\hat{C}_e = -\log(1 - \hat{\Psi}_e)$;
  Simulate running through the $\hat{\Psi}$-optimal path from $\boldsymbol{v}_0$ using the actual Ψ;
  Update ϕ_e and N_e accordingly;
end while

Unfortunately, as we show in Section 5, the performance of such methods is rather poor in our continuous setting. First, to ensure $\mathfrak{S}_j \to 0$, the number of actions/edges per node would have to grow as we refine the graph – otherwise, we will not be able to obtain all possible directions of motion in the limit. The methods above are hard to use in MDPs with large action sets and are not directly usable in MDPs with infinite action spaces. Second, in UCT any capture yields an equal penalty for all edges successfully traversed in that episode (even if the true Ψ_e is quite low along some of them). This is why model-based methods, such as Alg-D, are preferable for this class of problems even on graphs. Third, and most importantly, both UCT and Alg-D do not account for correlations in (unknown) transition functions of different (state, action) pairs. In the continuous case, the smoothness of surveillance rate $K(\boldsymbol{x})$ makes the spatial correlations crucial. Ignoring this feature results in much slower learning.

In the next section, we overcome these limitations by introducing new methods for *continuous model* learning and path planning based on a confidence-bounds-modified version of the model.

[3]In Supplementary Materials, we prove that under Alg-D the expected regret of a graph-restricted problem tends to 0 as the number of episodes tends to ∞.

4 MODEL-BASED METHODS ON Ω

4.1 Piecewise-constant models of K and planning based on confidence bounds

As a first attempt to solve the continuous problem, we decompose Ω into a collection $\mathcal{G}$ of non-overlapping subdomains/cells and assume that $K(\boldsymbol{x})$ is a constant on each of them. We define three auxiliary objects to gather data inside each cell over many episodes:

- $\mathcal{G}_c$: the total number of captures in a cell;
- $\mathcal{G}_t$: the total time spent in a cell;
- $\mathcal{G}_n$: the total number of visits/entries into a cell.

We will use $\tilde{K} \in R_{0,+}^{|\mathcal{G}|}$ to represent a piecewise constant estimate of $K(\boldsymbol{x})$ and $\tilde{\sigma}^2 \in R_{0,+}^{|\mathcal{G}|}$ to denote the element-wise estimated variance of $\tilde{K}$.

Focusing on a single cell, suppose that the surveillance intensity is indeed some (unknown) constant: $K(\boldsymbol{x}) = K_{\text{cell}}$. Enumerating all episodes in which our planned trajectory involved traveling through that cell and capture did not occur before we reached it, suppose in the k-th such episode our plan is to exit that cell after time t_k. The capture time S_k would be an exponentially distributed random variable with rate K_{cell}, but of course we only get to observe its *right censored* (Shorack and Wellner, 2009) version $R_k = \min(S_k, t_k)$. For convenience, we also define a *capture indicator* δ_k, which is equal to 1 if we are caught before exiting that cell and 0 otherwise (implying $R_k = t_k$). Assuming there were n such visits to this cell up to the current episode, this right-censored data $(\delta_k, R_k)_{k=1}^n$ can be used to derive the maximum likelihood estimate (MLE)

$$\tilde{K}_{\text{cell}} = \sum_{k=1}^{n} \delta_k \Big/ \sum_{k=1}^{n} R_k. \qquad (8)$$

The asymptotic expression for $\tilde{K}_{\text{cell}}$'s variance is

$$K_{\text{cell}}^2 \Big/ \sum_{k=1}^{n} [1 - \exp(-K_{\text{cell}} t_k)]. \qquad (9)$$

Using the fact that $\mathbb{E}[\delta_k] = 1 - \exp(-K_{\text{cell}} t_k)$, we can estimate this asymptotic variance as

$$\tilde{\sigma}_{\text{cell}}^2 = \sum_{k=1}^{n} \delta_k \Bigg/ \left(\sum_{k=1}^{n} R_k\right)^2. \qquad (10)$$

Using $\mathcal{G}_c$ and $\mathcal{G}_t$ notation, $\tilde{K}$ and $\tilde{\sigma}$ can be written as $\tilde{K} = \mathcal{G}_c / \mathcal{G}_t, \tilde{\sigma}^2 = \mathcal{G}_c / \mathcal{G}_t^2$ on each cell.

Inspired by the confidence bound techniques on graphs, we can also build up a "lower-confidence" intensity. Since this

modification can produce negative values and our observation intensity must be non-negative, we do not approximate K directly but instead model $Z(\boldsymbol{x}) = \log K(\boldsymbol{x})$. We use the statistic $\tilde{Z}_{\text{cell}} = \log(\tilde{K}_{\text{cell}})$ as an estimator for Z values at a cell center, with $\tilde{K}_{\text{cell}}$ defined as in (9). If $\tilde{K}_{\text{cell}}$ is asymptotically distributed as $N(\mu_K, \sigma_K^2)$ with σ_K approaching 0, then by local linearization of the logarithm (known as the *delta method* (Van der Vaart, 2000)), we have that $\tilde{Z}_{\text{cell}}$ is asymptotically distributed as $N(\log \mu_K, \sigma_K^2/\mu_K^2)$. Using $\mathcal{G}_c$ and $\mathcal{G}_t$ notation, we estimate the mean and variance for $\tilde{Z}_{\text{cell}}$ by

$$\tilde{Z} = \log\left(\mathcal{G}_c/\mathcal{G}_t\right), \quad \tilde{\sigma}_Z^2 = 1/\mathcal{G}_c. \tag{11}$$

This allows us to define corresponding piecewise-constant functions $\tilde{Z}(\boldsymbol{x})$ and $\tilde{\sigma}_Z(\boldsymbol{x})$. Our lower-confidence-adjusted intensity is constructed as

$$\hat{K}(\boldsymbol{x}) = \exp\left(\tilde{Z}(\boldsymbol{x}) - \sqrt{\log(T|\mathcal{G}|/\gamma)}\tilde{\sigma}_Z(\boldsymbol{x})\right). \tag{12}$$

Designed similarly to (7), the constant factor multiplying $\tilde{\sigma}_Z(\boldsymbol{x})$ balances the exploration vs exploitation and depends on the total number of episodes T, the number of cells $|\mathcal{G}|$, and $\gamma \in (0,1)$ controls the rate of exploration. The resulting formula yields low values of $\hat{K}$ in rarely visited cells, thus encouraging the exploration if those cells are relevant for paths from $\boldsymbol{x}_0$ to $\partial\Omega$. (E.g., a cell with a low $\hat{K}$ might be completely irrelevant if passing through it requires traversing other high-$\hat{K}$ cells or if there exists a much shorter/safer path from $\boldsymbol{x}_0$ to $\partial\Omega$.) Since the above defined $\hat{K}$ is also piecewise-constant, the $\hat{K}$-optimal trajectory will be polygonal, but finding its exact shape still requires solving an Eikonal equation. We use a Fast Marching Method (Sethian, 1996) to do this on a finer grid $\mathcal{G}_{\text{pd}}$.

Algorithm 3 summarizes the resulting method. To avoid $\tilde{Z}$ in (11) becoming infinity, we initialize $\mathcal{G}_t$ as a small positive constant ϵ and $\mathcal{G}_c$ as ϵK_{init} so that initially $\mathcal{G}_c/\mathcal{G}_t$ equals to some constant K_{init}[4].

4.2 GP regression models of K and planning based on confidence bounds

Algorithm 3 ignores the correlations between K values in different cells and updates each cell independently. In this section, we capture spatial correlation in the intensity by using a Gaussian process (GP) (Williams and Rasmussen, 2006). A Gaussian process is a collection of Gaussian random variables indexed by $\boldsymbol{x}$, with mean $m(\boldsymbol{x})$ and covariance $\Sigma(\boldsymbol{x}, \boldsymbol{x}')$. Some common choices of $\Sigma(\boldsymbol{x}, \boldsymbol{x}')$ are, e.g. the squared exponential kernel

$$\Sigma(\boldsymbol{x}, \boldsymbol{x}') = \alpha \exp\left(-|\boldsymbol{x} - \boldsymbol{x}'|^2/\beta^2\right) \tag{13}$$

[4]In our numerical tests, we chose K_{init} to be the Ω-averaged value of K. We have also experimented with other values of K_{init}, but this did not seem to have significant effect on Algorithm's performance.

Algorithm 3 Alg-PC: planning with a piecewise-constant model

Input: γ, ϵ.
Set $\mathcal{G}_c = \epsilon K_{\text{init}}$, $\mathcal{G}_t = \epsilon$;
Set $\tilde{Z} = \log(\mathcal{G}_c/\mathcal{G}_t)$, $\tilde{\sigma}_Z = 1/\mathcal{G}_c$;
while $t = 1 : T$ **do**
  $\hat{K}(\boldsymbol{x}) = \exp\left(\tilde{Z}(\boldsymbol{x}) - \sqrt{\log(T|\mathcal{G}|/\gamma)}\tilde{\sigma}_Z(\boldsymbol{x})\right)$;
  Numerically solve $|\nabla \hat{u}(\boldsymbol{x})| = \hat{K}(\boldsymbol{x})$;
  Find an optimal path from $\boldsymbol{x}_0$ using $\hat{u}(\boldsymbol{x})$;
  Simulate that path using the real $K(\boldsymbol{x})$;
  Update $\mathcal{G}_c, \mathcal{G}_t, \mathcal{G}_n$ using simulation results;
  Compute $\tilde{Z}, \tilde{\sigma}_Z$ according to (11);
end while

or the Matérn kernel

$$\Sigma(\boldsymbol{x}, \boldsymbol{x}') = \alpha \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} d/\beta\right)^{\nu} B_\nu \left(\sqrt{2\nu} d/\beta\right) \tag{14}$$

where $d = |\boldsymbol{x} - \boldsymbol{x}'|$, Γ is the gamma function, and B_ν is the modified Bessel function of the second kind. Here, (α, β) are hyperparameters that can be learned from data while the parameter ν in the Matérn kernel controls the differentiability of GP and can reflect our assumptions about the level of regularity of $K(\boldsymbol{x})$.

Criteria*. We only use cells whose estimates of K are accurate enough as inputs of GP regression. Here we introduce a list of rules to select these cells from $\mathcal{G}$:

- $\mathcal{G}_c \geq 1$, preventing $1/\mathcal{G}_c$ from becoming infinity.
- $\mathcal{G}_n \geq n_{\min}$, guaranteeing enough entries into a cell. Our implementation uses $n_{\min} = 20$.
- $\mathcal{G}_t \geq t_{\min}$, avoiding extremely short traverses. Our implementation sets $t_{\min}$ to the time sufficient to traverse a cell's diameter.

We denote the cells satisfying **Criteria*** as $\mathcal{G}_{\text{ob}} \subset \mathcal{G}$ and their centers as X_{ob}. Let $\tilde{Z}_{\text{ob}}, \tilde{\sigma}_{\text{ob}}$ be $\tilde{Z}, \tilde{\sigma}_Z$ values at X_{ob} reshaped as vectors. Use $\tilde{\Sigma}$ as an abbreviation of $[\Sigma_{\text{ob}} + \text{diag}(\tilde{\sigma}_{\text{ob}})]$. The GP posterior mean $M(\boldsymbol{x})$ for Z based on noisy observations at X_{ob} is

$$M(\boldsymbol{x}) = m(\boldsymbol{x}) + \Sigma(\boldsymbol{x}, X_{\text{ob}})\tilde{\Sigma}^{-1}\left[\tilde{Z}_{\text{ob}} - m(X_{\text{ob}})\right]. \tag{15}$$

Another advantage of GP model is that we simultaneously obtain the posterior covariance

$$\rho^2(\boldsymbol{x}) = \Sigma(\boldsymbol{x}, \boldsymbol{x}) - \Sigma(\boldsymbol{x}, X_{\text{ob}})\tilde{\Sigma}^{-1}\Sigma(X_{\text{ob}}, \boldsymbol{x}). \tag{16}$$

The resulting method is summarized in Algorithm 4. We note that partial knowledge $K(\boldsymbol{x})$ can be encoded in a prior mean $m(\boldsymbol{x})$, though in our experiments $m(\boldsymbol{x})$ was simply set to a positive constant on the entire Ω.

Algorithm 4 Alg-GP: planning with a GP model

Input: $m(\boldsymbol{x}), (\alpha, \beta), \gamma$
  Set $\mathcal{G}_c, \mathcal{G}_t, \mathcal{G}_n$ to be all zeros;
  Choose a kernel $\Sigma(\boldsymbol{x}, \boldsymbol{x}')$;
  $M(\boldsymbol{x}) = m(\boldsymbol{x}), \rho(\boldsymbol{x}) = 0$;
  while $t = 1 : T$ **do**
    $\hat{K}(\boldsymbol{x}) = \exp(M(\boldsymbol{x}) - \sqrt{\log(T|\mathcal{G}|/\gamma)}\rho(\boldsymbol{x}))$;
    Numerically solve $|\nabla \hat{u}(\boldsymbol{x})| = \hat{K}(\boldsymbol{x})$;
    Find an optimal path from $\boldsymbol{x}_0$ using $\hat{u}(\boldsymbol{x})$;
    Simulate that path using the real $K(\boldsymbol{x})$;
    Update $\mathcal{G}_c, \mathcal{G}_t, \mathcal{G}_n$ using simulation results;
    Compute $\tilde{Z}, \tilde{\sigma}_Z$ according to (11);
    Determine $\mathcal{G}_{\text{ob}}$ according to **Criteria***;
    if $\mathcal{G}_{\text{ob}}$ is non-empty **then**
      Update $M(\boldsymbol{x})$ and $\rho(\boldsymbol{x})$ using (15) and (16);
    end if
    if $t > 1000$ and $t \equiv 1$ mod(1000) **then**
      Tune hyperparameters (α, β) according to (17).
    end if
  end while

4.3 Hyperparameter tuning

The values of (α, β) are essential to the performance of GP regression. GP provides a probabilistic framework for automatically selecting appropriate hyperparameters through maximizing log marginal likelihood (Dong et al., 2017):

$$\max_{\alpha,\beta>0} \; -\frac{1}{2}\boldsymbol{z}_{\text{ob,c}}^{\intercal}\tilde{\Sigma}^{-1}\boldsymbol{z}_{\text{ob,c}} - \frac{1}{2}\log|\tilde{\Sigma}| - \frac{n}{2}\log 2\pi, \quad (17)$$

where $\boldsymbol{z}_{\text{ob,c}} = \tilde{Z}_{\text{ob}} - m(X_{\text{ob}})$ is the vector of centered observations. In Algorithm 4, we conduct this hyperparameter tuning once every thousand episodes.

5 NUMERICAL EXPERIMENTS

We now compare the performance of Algorithms UCT, Alg-D, Alg-PC, and Alg-GP on several examples[5] on the domain $\Omega = [0, 1]^2$. For simplicity, we assume $f(\boldsymbol{x}) = 1$ and focus on different versions of $K(\boldsymbol{x})$, constructed as sums of several Gaussian peaks with different amplitudes and widths. (Each peak might correspond to the location of a separate observation/surveillance center.) For each example, we conduct $T = 15000$ episodes, always starting from the same initial position $\boldsymbol{x}_0$ indicated by a cyan dot.

In all examples, we have benchmarked UCT on a graph G built on a 20×20 grid of nodes. The same graph was also used to test the performance of Alg-D. In Alg-PC and Alg-GP, all examples use a 20×20 observation grid $\mathcal{G}$ while the Eikonal equations are solved on a 101×101 grid $\mathcal{G}_{\text{pd}}$. In all cases, we have used $\lambda = \sqrt{2}$ and $\gamma = 0.1$.

Figures 1-3 provide detailed information on three representative examples, focusing on Alg-GPe (a version of Alg-GP using the squared exponential kernel). In each case, the four subfigures (enumerated left to right, top to bottom) present: (i) The true surveillance intensity $K(\boldsymbol{x})$. (ii) The level sets of $u(\boldsymbol{x})$; i.e., the minimum integral of K along a trajectory starting at $\boldsymbol{x}$. (iii) The final $\exp(M(\boldsymbol{x}))$; i.e., the Alg-GP-predicted $K(\boldsymbol{x})$ without confidence bound modification. The magenta dots are locations of captures (only from experiments of Alg-GP). The black curve is the optimal path based on $\exp(M(\boldsymbol{x}))$ after the last episode. (However, note that in each episode the trajectory is planned according to a version of $\hat{K}(\boldsymbol{x})$ available at that time.) (iv) The final GP posterior variance $\rho(\boldsymbol{x})$. Figure 4 shows the regret metric (i.e., the excess rate of captures $\mathfrak{S}$) for all of the benchmarked algorithms.

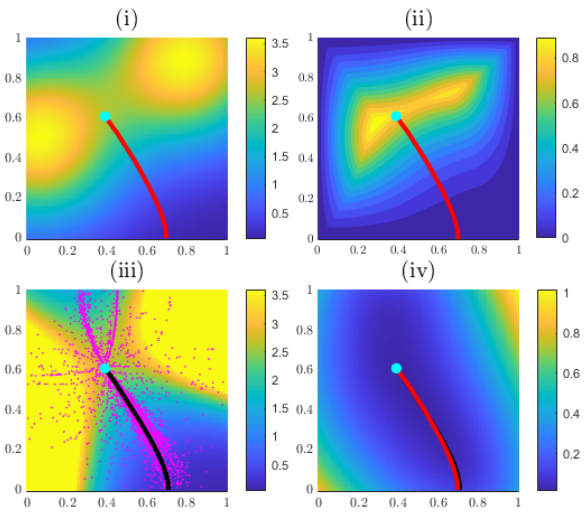

Figure 1: Bimodal surveillance intensity $K(\boldsymbol{x})$. Most of the selected paths are around three locally optimal paths with the longer one being globally optimal.

The following observation hold true for all examples:

- Capture locations (magenta dots) in subfigure (iii) give rough indications of which parts of Ω Alg-GP prefers to explore. It mostly selects paths around locally optimal ones, and during the later episodes it focuses more on the vicinity of the globally optimal one, showing a proper balance between exploration and exploitation. Indeed, the two paths in subfigure (iv) indicate that Alg-GP's final prediction (black) approximately matches the true optimal path

[5] In the interest of computational reproducibility, the source code of our implementation and additional experiments can be found at `https://eikonal-equation.github.io/Bayesian-Surveillance-Evasion/`. GP updates, capture event simulations and the main loop of Alg-GP all Algorithms are implemented in MATLAB while Fast Marching Method and the optimal path tracer are in C++. We ran the experiments using Dell OptiPlex 7050 desktop with 3.6 GHz Intel i7-7700 processor, and 16 GB RAM. With a 20×20 observation grid, each of these examples took under 20 minutes with Alg-GP and under 5 minutes with Alg-PC.

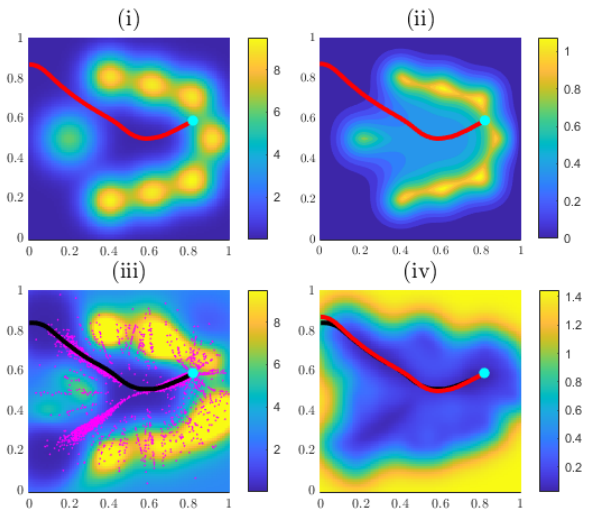

Figure 2: Surveillance intensity inspired by Figure 7 from (Cartee and Vladimirsky, 2020). The shortest exiting path induces a higher capture probability, while the actual optimal path takes a longer detour towards a lower intensity region. Selected paths mostly cluster around two locally optimal paths, with one of them being globally optimal.

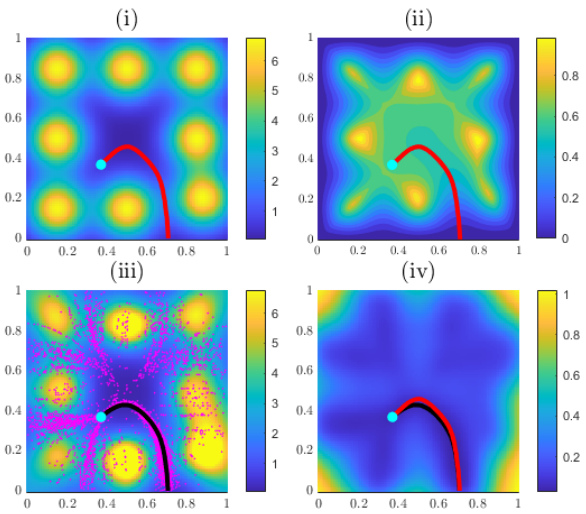

Figure 3: An intensity with eight peaks and multiple locally optimal paths. The peak around the southeast corner is displaced slightly, creating a gap in $K(\boldsymbol{x})$ and the optimal path reflects this. Alg-GP attempts to discover each locally optimal path but concentrates more around three paths with one of them being globally optimal.

(red).

- In Figure 4, Alg-GP's averaged excess capture rate continuously decreases and appears to confirm the convergence. The third example contains multiple locally optimal (and roughly comparable) paths and requires more episodes to learn the globally optimal one, which explains the slow decrease of $\mathfrak{S}$ in later episodes.
- In contrast, Alg-D and Alg-PC exhibit a consistently slower improvement of $\mathfrak{S}$ and even stagnation – indicating that model-learning takes substantially longer for these methods.
- UCT algorithm generates much larger regrets than all others. We have also tested it with more episodes ($10^5 \sim 10^6$), but the results still show obvious stagnation. The main reason is that UCT learns the state-action functions directly. It regards a whole path as a single datum and overlooks the information of not being captured along the earlier parts of the path, which could be used to improve the estimates for K.
- Both Alg-D and UCT restrict the Evader to move only along one of the eight directions. In principle, this will result in a gap between the truly optimal W_* and the risk along the best path on this graph even after infinitely many episodes. However, as we show in Supplementary Materials, in these examples that gap is much smaller than the observed regret $\mathfrak{S}$.

Remarks on grid effects: Our use of a discrete observation grid $\mathcal{G}$ essentially lumps together all captures within one cell and limits the algorithm's ability to learn the true K. To account for this more accurately, we could modify our definition of the regret metric, measuring the regret relative to the best path learnable on the specific $\mathcal{G}$. We include such detailed tests in Supplementary Materials. But in summary we note that they demonstrate yet another advantage of GP regression:

1) For Alg-GP, if we assume infinitely many captures in each cell, the ideal (asymptotically learnable) $\tilde{W}_* = 1 - \exp(\mathcal{J}_*)$ is only very weakly dependent on $\mathcal{G}$. E.g., the $\tilde{W}_*$ is already within 0.25% from the truly optimal $W_* = 1 - \exp(-u(\boldsymbol{x}_0))$ for all examples considered above even with a coarse 10×10 observation grid.

2) For Alg-PC, the ideal learnable $\tilde{W}_*$ can improve significantly when we refine $\mathcal{G}$. Asymptotically, the averaged regret is certainly better for finer observation grids. But they also present a challenge since many more episodes are needed to obtain a reasonable approximation of $\tilde{K}$ in all potentially relevant cells before the finer grid's asymptotic advantage becomes relevant. E.g., re-running the example from Figure 2 with $T = 60,000$, Alg-PC yields a lower averaged regret on a 10×10 observation grid than on a 40×40 grid for the first 50,000 episodes, while the averaged regret on a 20×20 grid is much smaller throughout.

3) The effects of the computational grid $\mathcal{G}_{\text{pd}}$ appear to be negligible in all of our simulations. The errors due to discretizing the Eikonal PDE are dominated by the errors due to uncertainty in K and the $\mathcal{G}$ effects described above.

Computational complexity: The cost of each episode consists of two components: (a) updating surveillance intensity model and (b) HJB-based path-planning using the

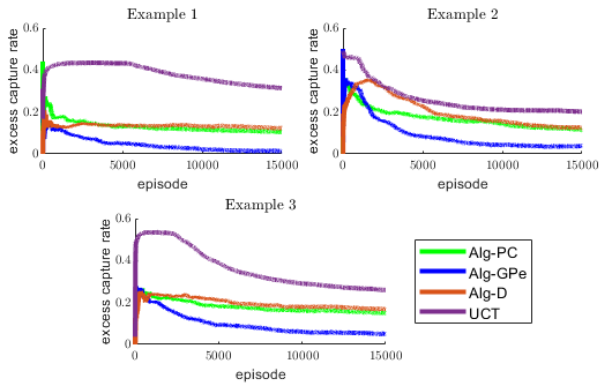

Figure 4: Regret metric (excess capture rate $\mathfrak{S}$) for examples illustrated in Figures 1, 2 and 3.

current model. The cost of (a) for Alg-PC scales as $O(|\mathcal{G}|)$; i.e., linearly in the number of subdomains/cells. For Alg-GP, the complexity of this stage depends on the size of $\mathcal{G}_{ob}$ (the collection of cells satisfying **Criteria***). During each episode, the cost of Alg-GP's update is dominated by solving the linear system in updating posterior covariance (16), which is $O(|\mathcal{G}_{ob}|^3)$ and $|\mathcal{G}_{ob}|$ is usually quite small compared to $|\mathcal{G}|$. Alg-GP also includes an additional cost of periodically re-tuning the hyperparameters.

The use of the Fast Marching Method makes the actual path planning in each episode quite fast even for much finer $\mathcal{G}_{\text{pd}}$. The cost of (b) is dominated by solving the PDE on a grid, which in our setting is $O(N \log N)$ on a discretization grid $\mathcal{G}_{\text{pd}}$ with N gridpoints. The latter cost can be prohibitive in high-dimensional generalizations since N grows exponentially with the dimension of Ω. The usual approach to overcome this "curse of dimensionality" is to use Approximate DP (e.g., based on mesh-free HJB discretizations), but this is not necessary in our 2D setting.

Learning less regular K: We have also used Examples 1-3 to test Alg-GP with the Matérn kernel, taking $\nu = 5/2$. The results are quite similar to those of Alg-GPe and thus omitted. But the situation is noticeably different when the actual $K(\boldsymbol{x})$ is non-smooth or even discontinuous. Figure 5 shows two such examples, modifying the smooth intensity from Figure 3. Suppose the center of a peak is $\boldsymbol{x}_c$, our first example replaces each single Gaussian peak by $K_{\text{single}}(\boldsymbol{x}) = \max\{0,\, K_p - |\boldsymbol{x} - \boldsymbol{x}_c|\}$ where $K_p > 0$ is the maximum. Such a surveillance intensity (used in Example 4) is still continuous but not smooth. In Example 5, we replace each Gaussian peak by a piecewise constant function: $K_{\text{single}}(\boldsymbol{x}) = K_c$ when $|\boldsymbol{x} - \boldsymbol{x}_c| \le r$ for some radius $r > 0$ while $K_{\text{single}}(\boldsymbol{x}) = 0$ otherwise. In both cases, a small positive constant was later added to ensure that the resulting $K(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in \Omega$.

We have conducted these experiments with both the squared exponential kernel (Alg-GPe) and Matérn kernel (Alg-GPm) with $\nu = 1/2$. In the continuous but non-smooth Example 4, Alg-GPe still remains the best though Alg-GPm is almost as good. In the discontinuous Example 5, Alg-PC actually initially outperforms both GP-based algorithms though Alg-GPm eventually catches up.

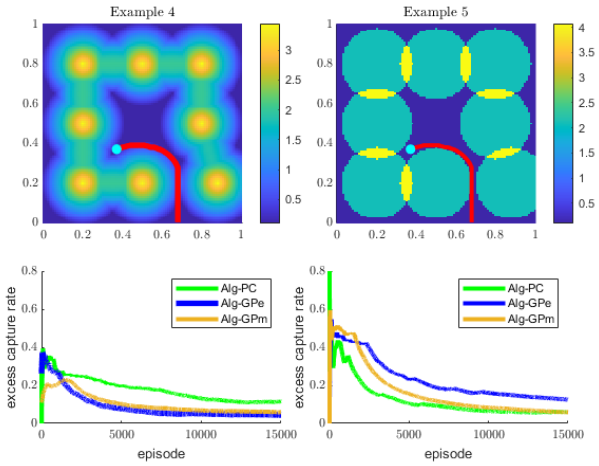

Figure 5: Two examples of non-smooth surveillance intensity. Example 4 (LEFT): a non-smooth-yet-continuous $K(\boldsymbol{x})$. Example 5 (RIGHT): a discontinuous $K(\boldsymbol{x})$. The Matérn kernel with $\nu = 1/2$, which assumes less smoothness of the interpolated function, performs better than the squared exponential kernel.

6 CONCLUSIONS

We developed and numerically tested three algorithms (Alg-D, Alg-PC, and Alg-GP) for continuous path-planning problems with unknown random termination/capture intensity. These algorithms follow a Bayesian approach to model the surveillance intensity K and then apply confidence bound techniques to tackle the exploration-exploitation dilemma. The GP-regression used in Alg-GP leverages the spatial correlations in K and usually results in more efficient learning from captures – particularly when the GP covariance kernel is chosen to reflect the smoothness of the actual K. While our experimental results are very promising, we do not currently have a proof of convergence and rigorous upper bound on the cumulative regret. We hope that the graph-theoretic UCB proofs (Azar et al., 2017) can be extended to cover edge-correlations and ultimately to our continuous setting.

Several extensions would broaden the applicability of our approach. If we consider E's post-detection planning, the terminal cost will become spatially inhomogeneous (Andrews and Vladimirsky, 2014). Multiobjective control methods (Kumar and Vladimirsky, 2010) and various robust path planning techniques (Qi et al., 2021) will become relevant if the risk of detection is balanced against other

optimization criteria (e.g., the profit from smuggling resources from a protected area (Arnold et al., 2019; Cartee and Vladimirsky, 2020; Chen et al., 2021)). We have focused on the isotropic dynamics primarily to simplify the exposition. Non-isotropic controlled dynamics lead to more general HJB PDEs, which can be similarly solved on a grid to approximate the value function efficiently (Sethian and Vladimirsky, 2001; Tsai et al., 2003; Alton and Mitchell, 2012; Mirebeau, 2014). The gradient of that value function can be then used to synthesize the optimal control. If K is viewed as changing in time, this will introduce an additional challenge of change point detection (Aminikhanghahi and Cook, 2017) in our RL algorithms. It will be also interesting to consider an antagonistic version, where K is chosen by surveillance authorities to maximize the probability of capture in response to E's path choices. A Nash equilibrium for this problem was already studied under the assumption that K is selected (perhaps probabilistically) from a finite list of options $K_1, ..., K_r$, all of which are known to E (Gilles and Vladimirsky, 2020; Cartee et al., 2019). We hope that it can be also extended to our setting where K_i's are learned online.

Acknowledgements

This research was supported in part by the NSF DMS (awards 1645643, 1738010, and 2111522). The authors are also grateful to M. Wegkamp and M. Nussbaum for advice on handling the right-censored data.

References

Agarwal, A., Jiang, N., and Kakade, S. M. (2019). Reinforcement learning: Theory and algorithms.

Alton, K. and Mitchell, I. M. (2012). An Ordered Upwind Method with Precomputed Stencil and Monotone Node Acceptance for Solving Static Hamilton-Jacobi Equations. *Journal of Scientific Computing*, 51(2):313–348.

Aminikhanghahi, S. and Cook, D. J. (2017). A survey of methods for time series change point detection. *Knowledge and Information Systems*, 51(2):339–367.

Andrews, J. and Vladimirsky, A. (2014). Deterministic control of randomly-terminated processes. *Interfaces and Free Boundaries*, 16(1):1–40.

Arnold, D. J., Fernandez, D., Jia, R., Parkinson, C., Tonne, D., Yaniv, Y., Bertozzi, A. L., and Osher, S. J. (2019). Modeling environmental crime in protected areas using the level set method. *SIAM Journal on Applied Mathematics*, 79(3):802–821.

Auer, P. (2002). Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422.

Azar, M. G., Osband, I., and Munos, R. (2017). Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning*, pages 263–272. PMLR.

Bardi, M. and Capuzzo-Dolcetta, I. (2008). *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Springer Science & Business Media.

Cartee, E., Lai, L., Song, Q., and Vladimirsky, A. (2019). Time-dependent surveillance-evasion games. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 7128–7133. IEEE.

Cartee, E. and Vladimirsky, A. (2020). Control-theoretic models of environmental crime. *SIAM Journal on Applied Mathematics*, 80(3):1441–1466.

Chen, B., Peng, K., Parkinson, C., Bertozzi, A. L., Slough, T. L., and Urpelainen, J. (2021). Modeling illegal logging in Brazil. *Research in the Mathematical Sciences*, 8(2):1–21.

Dann, C. and Brunskill, E. (2015). Sample complexity of episodic fixed-horizon reinforcement learning. *Advances in Neural Information Processing Systems*, 28.

Dobbie, J. (1966). Solution of some surveillance-evasion problems by the methods of differential games. In *Proceedings of the 4th International Conference on Operational Research, MIT, John Wiley and Sons, New York, New York*.

Dong, K., Eriksson, D., Nickisch, H., Bindel, D., and Wilson, A. G. (2017). Scalable log determinants for Gaussian process kernel learning. In *Proceedings of NIPS 2017*.

Gilles, M. A. and Vladimirsky, A. (2020). Evasive path planning under surveillance uncertainty. *Dynamic Games and Applications*, 10(2):391–416.

Kocsis, L. and Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*, pages 282–293. Springer.

Kumar, A. and Vladimirsky, A. (2010). An efficient method for multiobjective optimal control and optimal control subject to integral constraints. *Journal of Computational Mathematics*, pages 517–551.

Lewin, J. and Olsder, G. J. (1979). Conic surveillance evasion. *Journal of Optimization Theory and Applications*, 27(1):107–125.

Mirebeau, J.-M. (2014). Efficient fast marching with Finsler metrics. *Numerische Mathematik*, 126(3):515–557.

Qi, D., Dhillon, A., and Vladimirsky, A. (2021). Optimality and robustness in path-planning under initial uncertainty. *arXiv preprint arXiv:2106.11405*.

Recht, B. (2019). A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:253–279.

Sethian, J. A. (1996). A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595.

Sethian, J. A. and Vladimirsky, A. (2001). Ordered upwind methods for static Hamilton–Jacobi equations. *Proceedings of the National Academy of Sciences*, 98(20):11069–11074.

Shorack, G. R. and Wellner, J. A. (2009). *Empirical processes with applications to statistics*. SIAM.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

Takei, R., Tsai, R., Zhou, Z., and Landa, Y. (2014). An efficient algorithm for a visibility-based surveillance-evasion game. *Communications in Mathematical Sciences*, 12(7):1303–1327.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.

Tsai, Y.-H. R., Cheng, L.-T., Osher, S., and Zhao, H.-K. (2003). Fast sweeping algorithms for a class of Hamilton–Jacobi equations. *SIAM J. Numer. Anal.*, 41(2):673–694.

Van der Vaart, A. W. (2000). *Asymptotic statistics*, volume 3. Cambridge University Press.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3):279–292.

Williams, C. K. and Rasmussen, C. E. (2006). *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

Supplementary Materials

A PROOF OF UPPER BOUND FOR Alg-D

In this section we modify the upper bound proof of UCB-VI Algorithm (Agarwal et al., 2019) to develop similar upper bounds on expected regret for our Alg-D described in section 3.1. We use superscript t to indicate the corresponding episode that a quantity belongs to. Recall that Ψ_e denotes the true probability of capture when attempting to traverse an edge $e \in \mathcal{E}$ while $\tilde{\Psi}_e^t = \phi_e^t / N_e^t$ denotes our current best estimate for Ψ_e based on the attempted traversals of e and captures on e up to the episode t.

We assume that there exist constants $\underline{\Psi}$ and $\overline{\Psi}$ such that $0 < \underline{\Psi} \le \Psi_e \le \overline{\Psi} < 1$ and $-\log(1-x)$ is L-Lipschitz on $[\underline{\Psi}, \overline{\Psi}]$. Since $C_e = -\log(1-\Psi_e)$ and $\tilde{C}_e^t = -\log(1-\tilde{\Psi}_e^t)$, we know that

$$\mathbb{P}\big(|\tilde{C}_e^t - C_e| \ge L\epsilon\big) \le \mathbb{P}\big(|\tilde{\Psi}_e^t - \Psi_e| \ge \epsilon\big).$$

For the estimate $\tilde{\Psi}_e^t$, Hoeffding's inequality leads to

$$\mathbb{P}\big(|\tilde{\Psi}_e^t - \Psi_e| \ge \epsilon\big) \le 2\exp(-2N_e^t\epsilon^2),$$

where N_e^t is how many times edge e is visited up until the t-th episode. Choosing $\epsilon = \sqrt{\frac{\log(T|\mathcal{E}|/\gamma)}{N_e^t}}$, we obtain

$$\mathbb{P}\left(\left|\tilde{\Psi}_e^t - \Psi_e\right| \ge \sqrt{\frac{\log(T|\mathcal{E}|/\gamma)}{N_e^t}}\right) \le \frac{2\gamma^2}{T^2|\mathcal{E}|^2}.$$

Applying Boole's inequality over all edges and all episodes, we obtain a union bound

$$\mathbb{P}\left(\bigcup_{e\in\mathcal{E},t\le T}\left|\tilde{C}_e^t - C_e\right| \ge L\sqrt{\frac{\log(T|\mathcal{E}|/\gamma)}{N_e^t}}\right) \le \sum_{e\in\mathcal{E},t\le T}\mathbb{P}\left(\left|\tilde{C}_e^t - C_e\right| \ge L\sqrt{\frac{\log(T|\mathcal{E}|/\gamma)}{N_e^t}}\right)$$

$$\le \sum_{e\in\mathcal{E},t\le T}\mathbb{P}\left(\left|\tilde{\Psi}_e^t - \Psi_e\right| \ge \sqrt{\frac{\log(T|\mathcal{E}|/\gamma)}{N_e^t}}\right) \le \frac{2\gamma^2}{T|\mathcal{E}|}.$$

This equation measures the model error since it bounds the difference between the estimated cost with the true cost. With probability at least $(1 - 2\gamma^2/T|\mathcal{E}|)$, $|\tilde{C}_e^t - C_e| \le L\sqrt{\frac{\log(T|\mathcal{E}|/\gamma)}{N_e^t}}$ for any edge and any episode. Along the t-th episode's path p^t, triangle inequality yields

$$\left|\sum_{e\in p^t}\tilde{C}_e^t - \sum_{e\in p^t}C_e\right| \le \sum_{e\in p^t}\left|\tilde{C}_e^t - C_e\right| \le L\sqrt{\log(T|\mathcal{E}|/\gamma)}\sum_{e\in p^t}\frac{1}{\sqrt{N_e^t}}.$$

We denote the last term as Δ_{p^t}. As $\left|\hat{\Psi}_e^t - \tilde{\Psi}_e^t\right| \le \sqrt{\frac{\log(T|\mathcal{E}|/\gamma)}{N_e^t}}$, using the Lipschitz condition of $-\log(1-x)$ we obtain

$$\left|\hat{C}_e^t - \tilde{C}_e\right| \le L\sqrt{\frac{\log(T|\mathcal{E}|/\gamma)}{N_e^t}}.$$

Therefore, by triangle inequality we have

$$\left|\sum_{e\in p^t}\hat{C}_e^t-\sum_{e\in p^t}C_e\right|\le\left|\sum_{e\in p^t}\hat{C}_e^t-\sum_{e\in p^t}\tilde{C}_e^t\right|+\left|\sum_{e\in p^t}\tilde{C}_e^t-\sum_{e\in p^t}C_e\right|\le 2\Delta_{p^t}.$$

We denote v_{p^t} the true total cost of p^t (i.e., the sum of true edge costs C_e along p^t) and $\hat{v}_{p^t}$ the total cost based on the modified cost $\hat{C}_e$. Using $v_{p^t}=\sum_{e\in p^t}C_e$ and $\hat{v}_{p^t}=\sum_{e\in p^t}\hat{C}_e^t$, we finally obtain

$$\mathbb{P}\big(\forall t,|\hat{v}_{p^t}-v_{p^t}|\le 2\Delta_{p^t}\big)\ge 1-\frac{2\gamma^2}{T|\mathcal{E}|}.$$

Notice that either $\hat{C}_e^t=-\log(1-\underline{\Psi})\le C_e$ or $\hat{\Psi}_e^t=\tilde{\Psi}_e^t-\sqrt{\frac{\log(T|\mathcal{E}|/\gamma)}{N_e^t}}\le\Psi_e$ (hence $\hat{C}_e^t\le C_e$) with the probability of at least $q=(1-\frac{2\gamma^2}{T|\mathcal{E}|})$. Both cases lead to

$$v_{p^t}-2\Delta_{p^t}\le\hat{v}_{p^t}\le\hat{v}_{p^*}=\sum_{e\in p^*}\hat{C}_e^t\le\sum_{e\in p^*}C_e=v_{p^*},$$

where the second inequality follows from the $\hat{C}_e^t$-optimality of p^t while the first & third inequalities hold with probability of at least q.

Assuming that each path the algorithm considers has no more than $M\le|\mathcal{E}|$ edges, the expected cumulative regret can be bounded as

$$\begin{aligned}\mathbb{E}\left[\sum_{t=1}^{T}(v_{p^t}-v_{p^*})\right]&\le\mathbb{P}\big(\forall t,\,v_{p^t}-v_{p^*}\le 2\Delta_{p^t}\big)\sum_{t=1}^{T}(v_{p^t}-v_{p^*})+\mathbb{P}\big(\exists t,\,v_{p^t}-v_{p^*}>2\Delta_{p^t}\big)\sum_{t=1}^{T}M\overline{q}\\&\le 2L\sqrt{\log(T|\mathcal{E}|/\gamma)}\sum_{t=1}^{T}\sum_{e\in p^t}\frac{1}{\sqrt{N_e^t}}+\frac{2\gamma^2M\overline{C}}{|\mathcal{E}|},\end{aligned}$$

where $\overline{C}=-\log(1-\overline{\Psi})$.

Lemma 1. *The first term in previous inequality satisfies*

$$\sum_{t=1}^{T}\sum_{e\in p^t}\frac{1}{\sqrt{N_e^t}}\le\sqrt{MT|\mathcal{E}|}.$$

Proof. A different way to do the summation leads to

$$\sum_{t=1}^{T}\sum_{e\in p^t}\frac{1}{\sqrt{N_e^t}}=\sum_{e\in\mathcal{E}}\sum_{i=1}^{N_e^T}\frac{1}{\sqrt{i}}\le\sum_{e\in\mathcal{E}}\sqrt{N_e^T}\le\sqrt{|\mathcal{E}|}\sqrt{\sum_{e\in\mathcal{E}}N_e^T}\le\sqrt{MT|\mathcal{E}|}.$$

The first inequality is due to the Jensen's inequality while the second follows from the Cauchy-Schwartz inequality. The last one uses the fact that there are no more than MT visits to all the edges. □

Theorem 1. *Using Lemma 1, we can bound the averaged expected regret of our Alg-D on graph as*

$$\frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T}(v_{p^t}-v_{p^*})\right]\le\frac{2L}{T}\sqrt{2MT|\mathcal{E}|\log(T|\mathcal{E}|/\gamma)}+\frac{2\gamma^2M\overline{C}}{T|\mathcal{E}|}.\tag{18}$$

In particular, as $T\to\infty$, the averaged expected regret converges to zero.

We note that the above bounds apply to a graph-constrained problem only. The graph-optimal capture probability $(1-\exp[-v_{p^*}])$ might in principle be significantly larger than the $W_*=1-\exp[-u(\boldsymbol{x}_0)]$ describing the minimized capture probability in the continuous domain Ω.

B HYPERPARAMETER TUNING

We conduct hyperparameter tuning (17) every thousand episodes by applying the constrained optimizer `fmincon` with gradients in MATLAB . Assume the log marginal likelihood in (17) is $\log\left(\tilde{Z}_{\text{ob}}\middle|X_{\text{ob}},\alpha,\beta\right)$, the α-gradient can be computed as

$$\frac{\partial}{\partial\alpha}\log\left(\tilde{Z}_{\text{ob}}\middle|X_{\text{ob}},\alpha,\beta\right)=\frac{1}{2}\boldsymbol{z}_{\text{ob,c}}^{\intercal}\tilde{\Sigma}^{-1}\frac{\partial\tilde{\Sigma}}{\partial\alpha}\tilde{\Sigma}^{-1}\boldsymbol{z}_{\text{ob,c}}-\frac{1}{2}\text{tr}\left(\tilde{\Sigma}^{-1}\frac{\partial\tilde{\Sigma}}{\partial\alpha}\right). \tag{19}$$

$\boldsymbol{z}_{\text{ob,c}}^{\intercal}$ and $\tilde{\Sigma}$ are the same notations as defined in section 4.2. The expression of $\frac{\partial\tilde{\Sigma}}{\partial\alpha}$ depends on which kernel we choose. The β-gradient can be computed similarly.

C AVERAGED EXCESS RISK $\mathfrak{R}$

In this section, we define another path-related regret metric in addition to the excess rate of captures defined in (6). If $\boldsymbol{y}_i(\cdot)$ is the ith-episode path and $\mathcal{J}_i$ is the corresponding cumulative intensity, the capture probability along $\boldsymbol{y}_i(\cdot)$ is $W_i=1-\exp(-\mathcal{J}_i)$. The minimum capture probability is $W_*=1-\exp(-u(\boldsymbol{x}_0))$. The *averaged excess risk* is defined as

$$\mathfrak{R}_j=\frac{1}{j}\sum_{i=1}^{j}(W_i-W_*),\ j=1,2,\cdots,T. \tag{20}$$

Unlike excess rate of captures which uses probabilistic outcomes, $\mathfrak{R}$ compares directly the expected capture rate of episodic paths with the truly optimal W_*. Unfortunately, UCT algorithm is not guaranteed to find a path reaching the boundary during each episode. As a result, W_i cannot be computed and regret $\mathfrak{R}$ is not applicable to UCT. The following are $\mathfrak{R}$ results of Alg-D, Alg-PC and Alg-GP for all examples in this paper.

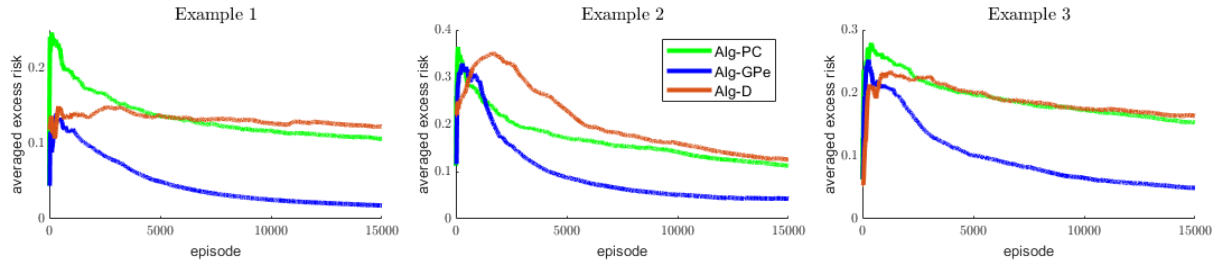

Figure 6: Regret metric (averaged excess risk $\mathfrak{R}$) for examples illustrated in Figures 1, 2 and 3.

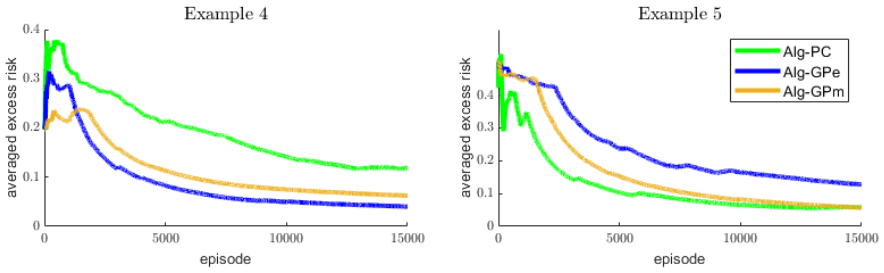

Figure 7: Regret metric (averaged excess risk $\mathfrak{R}$) for examples illustrated in Figures 5.

D LEARNABLE LIMITS RESTRICTED BY DISCRETIZATIONS

D.1 UCT and Alg-D

For graph algorithms UCT and Alg-D, the Evader selects paths only on the grid and is restricted to move only along eight fixed directions. As a result, even if we let the grid size approach zero, in the limit there is a gap between W_*

and the minimum capture rate that can be achieved on the grid. From the figure below we can observe, as the grid size decreases, the blue dots approach some level which is closer to the optimal (red line, original continuous case, computed on a $h = 1/800$ grid) but there remains a gap (≈ 0.014 when $h = 1/640$). Our numerical experiments use $h = 1/20$, in which case the best learnable limit is larger than the optimal capture probability by about 0.033. However, as we observe from Figure 2, after 15000 episodes the regret of Alg-D remains much larger than 0.033; i.e. the error due to this grid-restricted motion is not the reason why Alg-D shows such huge regrets. Using a smaller grid size can reduce this gap, but it also increases the number of parameters to be estimated. An overly fine grid takes more episodes to obtain accurate enough prediction of the capture rate, making it harder to observe its advantages.

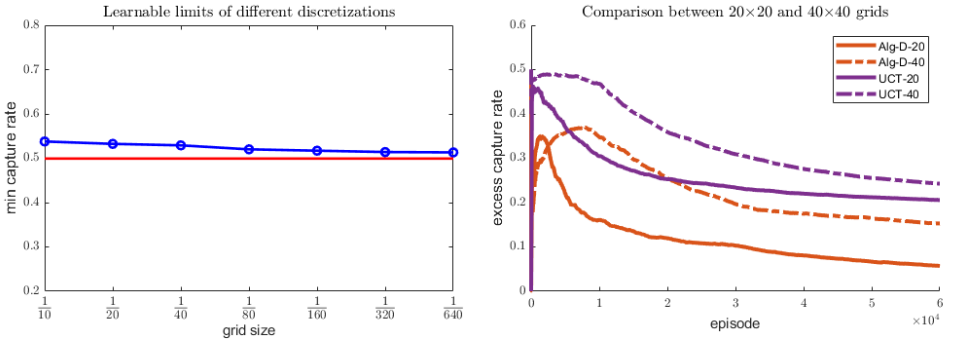

Figure 8: Left: the minimum capture probability over different discretized grids for the example in Figure 2. Right: We run UCT and Alg-D on 40×40 grids and compare regret $\mathfrak{S}$ with the results of 20×20 grids for 60000 episodes. We can observe from the figure that using a 20×20 grids generates much smaller regrets than 40×40.

D.2 Alg-PC and Alg-GP

We conduct an observation grid refinement study for Alg-PC using Example 2. The following figure shows the $\log$ of (non-averaged, instantaneous) differences between W_i and W_* in each episode, for different $\mathcal{G}$ resolutions. We observe that the lower $(W_i - W_*)$ values for 20×20 and 40×40 grids are much lower than for a 10×10 grid. This indicates that with a finer grid Alg-PC is able to explore paths which are closer to optimal.

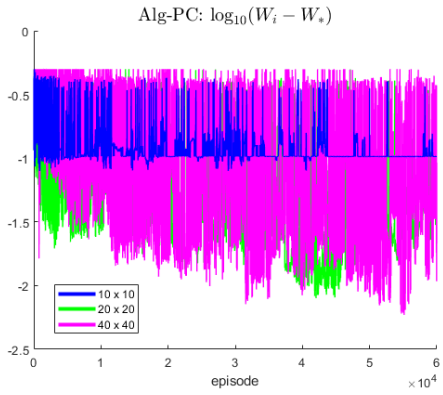

Figure 9: Instantaneous differences between W_i and W_* in each episode. The grids are $10 \times 10, 20 \times 20$ and 40×40.

Our use of the discrete observational grid $\mathcal{G}$ essentially lumps together all captures within one cell. This limits the algorithm's ability to learn the true K. To account for this more accurately, we can modify our definition of regret metric and measure the regret relative to the best path learnable on the specific $\mathcal{G}$. Below we provide additional tests illustrating this idea for Alg-PC and data showing that this subtlety is largely irrelevant for Alg-GP.

We define $\tilde{K}_*(\boldsymbol{x})$ to be the cell-averaged version of K. This piecewise-constant function also represents the best approximation of K that we could hope to attain with infinitely many captures in every cell. Computing the viscosity solution to $|\nabla \tilde{u}_*| = \tilde{K}_*$, we obtain the $\mathcal{G}$-optimal feedback control $\tilde{\boldsymbol{a}}_* = -\nabla \tilde{u}_* / |\nabla \tilde{u}_*|$ and can compute its actual

quality $w(\boldsymbol{x})$ by integrating the true K over the paths resulting from $\tilde{\boldsymbol{a}}_*$. The corresponding capture probability is then $\tilde{W}_* = 1 - \exp(-w(\boldsymbol{x}_0))$ and we can define the $\mathcal{G}$-adjusted regret by using $\tilde{W}_*$ instead.

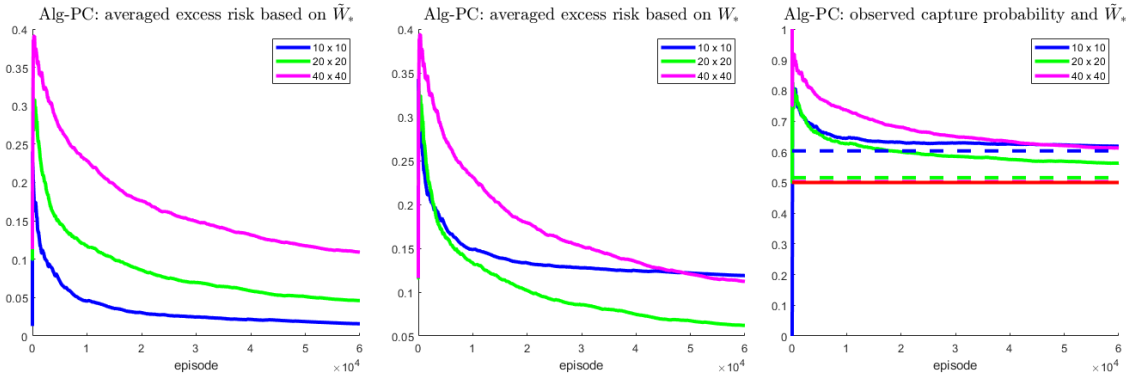

Figure 10: Left: the new regret defined as above for different sizes of $\mathcal{G}$, applying Alg-PC to Example 2. It illustrates that finer $\mathcal{G}$ requires more episodes to approach the asymptotic learnable limit. But even though the 20×20 $\mathcal{G}$ results in a higher grid-adjusted regret than the 10×10 grid, this is not the case for the regret-against-the-truly-optimal-W_*, as illustrated in the second (center) figure. The third (right) figure illustrates the experimentally observed capture rate (note that this is not the regret metric $\mathfrak{S}$), together with $\tilde{W}_*$ (dashed lines) for different sizes of $\mathcal{G}$. We observe that $\tilde{W}_*$ becomes closer to the true optimal W_* (the red horizontal line) as $\mathcal{G}$ becomes finer. But it takes about $50,000$ episodes until the asymptotic advantage of the 40×40 grid yields a lower capture rate than on the 10×10. In this example, the 20×20 grid yields much smaller regrets than the other two grids after $10,000$ episodes.

Gaussian process regression performs surprisingly well in recovering the optimal path, even on a coarse grid. For all examples considered in the paper, $\tilde{W}_*$ is already within 0.25% from the truly optimal W_* even with a coarse 10×10 observation grid. The following is an illustration of this phenomenon using the second and third examples from the paper.

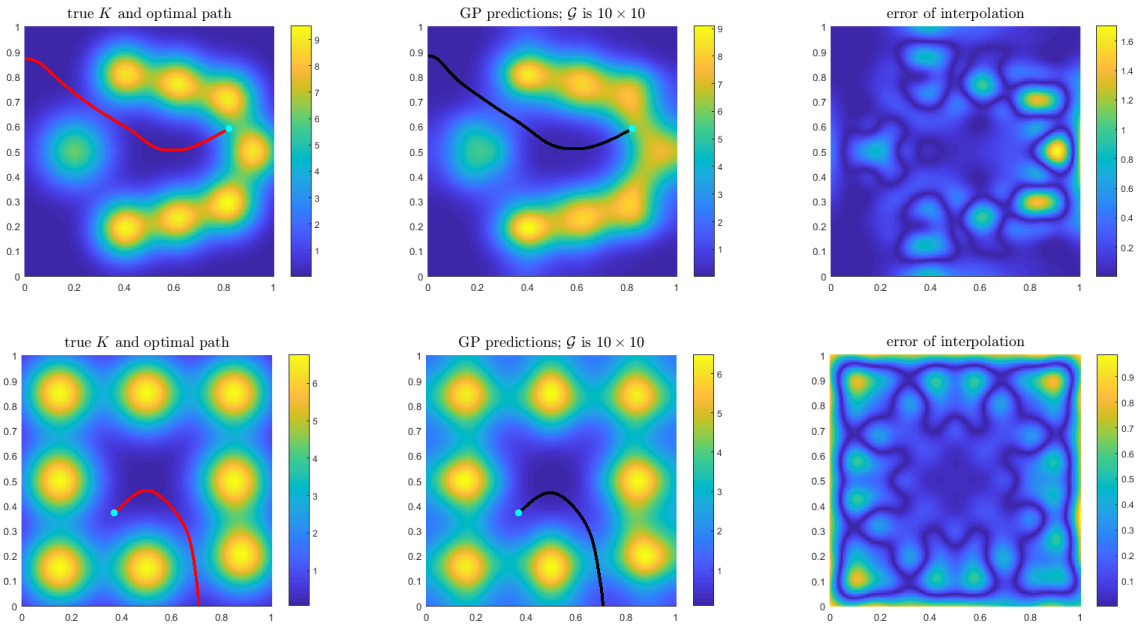

Figure 11: Left: true $K(\boldsymbol{x})$ and the truly optimal path. Center: GP-predicted intensity with an infinite number of observations on a 10×10 grid $\mathcal{G}$ and the corresponding "optimal" path. Right: interpolation errors. Top row: example 2; bottom row: example 3. In both cases, the GP-predicted optimal path is quite close to the true optimal path even though the GP-estimated $\tilde{K}_*$ is quite different from the true $K(\boldsymbol{x})$.