# Towards Robust Informative Path Planning for Spatiotemporal Environments

Srujan Deolasee

CMU-RI-TR-25-41

May 12, 2025


The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

**Thesis Committee:**
Prof. Katia Sycara, *chair*
Prof. Sebastian Scherer
Ananya Rao

*Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.*

*To my aai, baba, tai, aatya, aaji, and aaba – all of whom inspire me every day.*

# Abstract

Informative Path Planning (IPP) is an important planning paradigm for various real-world robotic applications such as wildfire monitoring and predicting infection spread in crops. IPP involves planning a path that can learn an accurate belief of the quantity of interest, while adhering to planning constraints. Traditional IPP methods are effective only in static, time-invariant environments and typically require high computation time during execution. This has given rise to reinforcement learning (RL) based IPP methods. However, existing RL-based methods do not address spatiotemporal environments, which present unique challenges due to variations in environment dynamics.

This thesis introduces a robust RL-based IPP framework specifically designed to enable robots to operate effectively across dynamic spatiotemporal environments. Our approach combines domain randomization with our proposed dynamics prediction model (DPM). The DPM constitutes a key contribution of our framework, explicitly modeling how environments evolve over time and extracting latent representations of their specific characteristics. Through extensive evaluations in a wildfire spread prediction task, we demonstrate that our DPM successfully infers environment dynamics online, enabling the RL policy to maintain consistent performance across environments with significantly different dynamics. The results suggest broad applicability to critical monitoring applications where environmental conditions vary considerably.

# Acknowledgments

I have been fortunate to embark on what has been a deeply rewarding journey at CMU. This thesis stands as a testament to the extraordinary people who have shaped this experience and made it truly meaningful.

First, I want to express my deepest gratitude to my advisor, Prof. Katia Sycara, whose wisdom and mentorship have been the cornerstone of my academic growth. Katia possesses that rare combination of brilliance and compassion that defines truly exceptional mentors and researchers. Katia gave me the freedom to explore many problems while continuously grounding our endless research discussions. Her passion for research, ability to identify meaningful problems, courage to question boldly, and persistence to explore uncharted research domains have left an indelible mark on me.

I am incredibly grateful to Siva Kailas, my PhD mentor, and Dr. Woojun Kim, my postdoc mentor. Siva brought an energy and passion to our research discussions that made even the most challenging problems feel like exciting adventures. His insights consistently opened new pathways of thinking. He always showed genuine care for my development as a researcher. Woojun's steady presence has been instrumental during my research journey. He has been a very important technical guide shaping this thesis. Both supported me through countless late-night paper deadlines, turning stressful moments into triumphs. My research contributions at CMU have been enriched by collaborations with some brilliant minds. Thanks to Prof. Wenhao Luo for continuously mentoring me remotely and always being available for guidance. Also thanks to Lingpeng, Srikar, and Emanuel—it was a pleasure to discuss research with all of you and work together on fascinating problems.

I would like to thank Prof. Sebastian Scherer for his guidance on high-level direction through our wildfire project meetings. Special thanks to Andrew Jong, the team lead, without whom none of this would have been possible. My sincere thanks to Ananya Rao for serving on my committee, and for her thoughtful feedback which has strengthened this thesis immeasurably.

Thanks to my labmates: Arjun, Benji, Dana, Emanuel, Joe, Karan, Muhan, Muyang, Renos, Sam, Sarthak, Sha Yi, Shreya, Silong, Simon, Yaqi, and Zifu. Special mention to Simon, my office companion who was always up for random life discussions, making the office feel warm.

My robotics journey began during my undergraduate years. Thanks to Aditya, Atharv, and Vedant for introducing me to this field. I am very grateful to Prof. John Dolan, Rachel Burcin, and Prof. Qin Lin for accepting me into the RISS program and as a UG thesis student. I am where I am because of them.

My time at CMU unfolded in two distinct chapters, each bringing friendships that became the highlight of my experience. During my initial journey as an undergraduate intern, I met a wonderful group of people who made my first stay outside India not just bearable, but truly enjoyable. I would like to thank my fellow RISS interns—Caleb, Jana, Larry, Rachel, Steven, Teodor, Vihaan, and Wenli—who taught me that friendship transcends all boundaries. I also want to thank the RI students who welcomed me with open arms and were always up for research discussions, life conversations, or simply a game of pool or table tennis. From the senior cohort: Nikhil, Nishant, Ravi, Sarvesh, Sashank, Seth, Shivesh, and Siddharth. From the incoming cohort: Aman, Anish, Bharath, Dvij, Jay, Nikhil, Prachi, Pranay, Pushkal, Sarthak, Shagun, Shreya, Sriram.

When I returned for my MSR degree, I was blessed to encounter another group of individuals who will hopefully stay lifelong friends: Aadesh, Aryan, Dhruv, Khush, Mehal, Parth, Peya, Rupali, Sagnik, Sanjali, Sashank, Shashwat, Shivangi, Tirtha, Vaidehi, Vihaan, and Yash. A special mention to my roommate, Neham, who was always there for me throughout MSR. His conversations helped me unwind after long, stressful days—and boy, there were many of those! I have countless precious memories with each one of them and will always cherish our time together. I could not have completed my MSR without their constant support and laughter. I would also like to thank my dear friends from undergrad for always keeping in touch despite the distance, and being available for those much-needed long video calls: Archit, Atharva, Siddharth, and Sujay.

Finally, I am very grateful to my family, whose love has been my anchor throughout this journey. Thank you aai (Sarika) for your endless sacrifices and unwavering belief in my dreams—you gave everything for my success. Thank you baba (Abhijit) for always being my pillar of strength, helping me get back up every time I stumbled and always encouraging me to enjoy the journey. Thank you tai (Sayali) for being my confidante and constant source of joy. My heartfelt thanks also go to my aatya, aaji, and aaba, whose wisdom has guided me every step of the way. All of you have helped shape me into a better person. Thank you for the boundless love.

## Funding

This work has been supported by NSF and USDA-NIFA under AI Institute for Resilient Agriculture, Award No. 2021-67021-35329 and Department of Agriculture Award Number 2023-67021-39073.

# Contents



*When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.*

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Informative path planning (IPP) has been actively studied for robotics deployments involving information acquisition, such as the autonomous exploration of unknown areas [5, 20], environment monitoring [11], and target tracking [40]. IPP aims to find a path for an autonomous robot that maximizes the acquisition of interests, e.g., the intensity of fire in fire monitoring, while adhering to resource constraints. Conventional IPP methods typically involve sampling-based path planning using a graph for static, time-invariant environments [1, 11, 17]. Recently, IPP solvers for spatio-temporal environments, where the interest changes over time, have also been proposed [13, 15]. These methods require heavy computation time to determine the path and are better suited for slower rate of spread of the environmental phenomenon (e.g., air temperature). This limits their applicability for real-world deployment, especially if the environment is changing rapidly (e.g., wildfire spread).

With the recent success of RL in various domains, RL-based IPP has been actively studied, demonstrating both superior performance and reduced computation time [4, 31, 39]. However, these methods have been developed for static environments like light-intensity variation [4], or fruit counting in orchards [39]. None of the prior works based on RL have considered spatio-temporal environments. There is an inherent problem of RL while trying to learn in such spatio-temporal environments: a lack of robustness against variations in environment dynamics [29]. That is, the agent demonstrates optimal performance only in the environment it was trained on, but not when environment dynamics vary. In addition, even if the agent is trained in

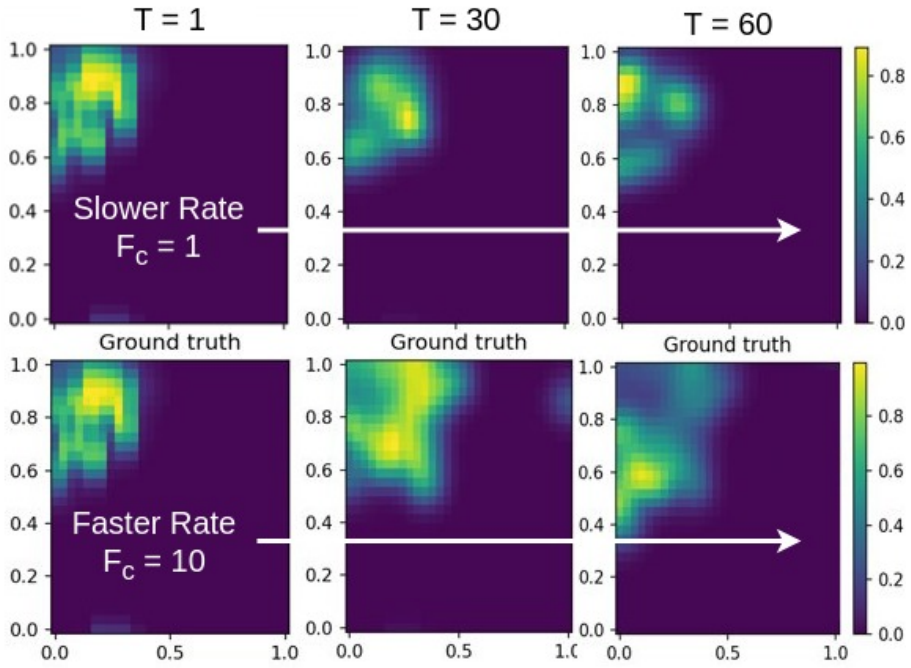


Figure 1.1: Consider a wildfire environment with two different dynamics: slow rate $F_c = 1$ (first row) and fast rate $F_c = 10$ (second row). $F_c$ is fuel coefficient in the FireCommander simulator [34] used to generate the wilfire evolution. Higher $F_c$ causes the fire to spread faster. The higher value (yellow) denotes a higher intensity of fire.

an environment with various dynamics, it exhibits suboptimal performance across different environment variations. To understand such problems regarding robustness, let us consider a wildfire domain, where the fire (interest) evolves with time. Here, the fire spread rate is determined by several characteristics, including fuel and wind velocity. Fig. 1.1 shows two environments with different fuel distributions, where the fuel parameter is denoted by $F_c$: $F_c = 1$ (first row) and $F_c = 10$ (second row). These environments have varying environment dynamics due to difference in fuels (1: slow, 10: fast). Consequently, the agent trained in the $F_c = 1$ environment has suboptimal performance in the $F_c = 10$ environment, and vice versa. The corresponding result in terms of the cumulative root mean squared error (RMSE) is shown in Table 1.1. The RMSE is computed between the environment prediction and the actual ground truth after every step of the agent's trajectory. Specifically, CAtNIPP[1] [4] trained

[1]This is a prior work on RL-based IPP. This will be discussed later.

| Model | Fuel coefficient | |
|---|---|---|
| | $F_c = 1$ | $F_c = 10$ |
| CAtNIPP (trained on $F_c$=1) | $13.7 \pm 2.6$ | $25.0 \pm 5.3$ |
| CAtNIPP (trained on $F_c$=10) | $16.2 \pm 3.6$ | $20.9 \pm 3.0$ |
| CAtNIPP + DR | $14.9 \pm 2.4$ | $24.0 \pm 5.6$ |
| `DyPNIPP` **(Ours)** | $\mathbf{10.3 \pm 1.6}$ | $\mathbf{14.7 \pm 2.9}$ |

Table 1.1: Performance comparison in terms of root mean squared error (RMSE) (lower is better) in the wildfire environment shown in Fig 1.1. The RMSE is computed between the environment prediction and the actual ground truth after every step of the agent's trajectory.

.

on $F_c = 10$ performs worse than that trained on $F_c = 1$ in the $F_c = 1$ environment, implying that the existing RL-based IPP algorithm lacks robustness against variations in environment dynamics.

In this paper, we propose a robust RL-based IPP framework named `DyPNIPP`, capable of operating effectively across environments with varying dynamics. The key insight is that robust performance requires both exposure to diverse conditions during training and the ability to recognize and adapt to specific conditions during deployment. Our approach addresses this fundamental challenge using two complementary strategies. We first train our agent across many different environmental scenarios - a technique known as domain randomization (DR) - which randomizes environment characteristics, e.g., fuel/vegetation distribution in the wildfire domain that determine the dynamics, enabling the agent to train across diverse set of environments. However, domain randomization alone is insufficient to train the policy to perform well in diverse environment dynamics. This is because the RL agent is likely trained for averaged dynamics and thus is not capable of inferring specific environment dynamics [37]. Therefore, we secondly introduce a dynamics prediction model that captures the environment dynamics *implicitly*, allowing the RL policy to recognize the current environment dynamics and adapt the policy accordingly.

In spatio-temporal environments, an IPP agent has to plan a path for an unknown future. A learned inferred dynamics model could greatly improve the capability of the agent to plan informative paths that are robust to unknown variations in the environment. For example, in a wildfire scenario with highly flammable fuel, an

agent must spend more time exploring to monitor rapid fire spread, whereas in low-flammability areas, the agent can focus on a more localized area. This adaptability is crucial for an IPP policy as it allows the agent to take actions suited to specific environmental dynamics. We evaluate `DyPNIPP` in the wildfire simulation environment with multiple environmental variations, such as the fuel coefficient affecting fire spread speed and the number of fire occurrences, as well as in an air temperature prediction environment. As seen in Table 1.1, the results show that domain randomization alone is not sufficient to robustify an RL policy; however, `DyPNIPP` improves the robustness of the RL policy for the spatio-temporal monitoring task.

The main contributions of this work are summarized as follows: (i) To the best of our knowledge, this is the first work addressing the robustness of RL-based informative path planning in spatio-temporal environments. (ii) We consider key factors for robustness in the wildfire domain, including fuel, vegetation coefficients, and the number of fires. (iii) We demonstrate the effectiveness of our approach across varying conditions—highlighting robustness—and provide analyses of `DyPNIPP`, along with a real-robot experiment showing the trained model's practical deployment.

# Chapter 2

# Background

## 2.1 Reinforcement Learning

RL aims to train an agent by interacting with an environment, and the decision making procedure is typically formulated as a Markov decision process (MDP) defined as the tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma >$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the transition probability, $r$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. At each time step $t$, the agent generates an action $a_t \in \mathcal{A}$ from a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ based on the given state $s_t \in \mathcal{S}$. The environment then yields the reward $r_t = r(s_t, a_t)$ and the next state $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$. The goal is to learn a policy that maximizes the expected discounted return, $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$. One representative algorithm is proximal policy optimization (PPO) [32]. PPO consists of an actor and a value function (critic), which estimates expected returns. The actor collects experience, and the value function estimates advantages, indicating how much better or worse an action is compared to the current policy expectation. The actor is then updated to maximize these advantages. PPO uses a clipping method to limit how much the new policy can deviate from the previous one, ensuring stable and efficient learning.

RL has been successfully applied to various robotics tasks, including locomotion [23] due to its decision-making capabilities. Our work focuses on informative path planning by leveraging the capabilities of RL and addressing its weaknesses in robustness, to achieve a solution that is both effective and robust.

## 2.2 Robustness

Robustness is considered an essential element for machine learning models to be practical solutions. Robustness can have different definitions depending on the context, such as resistance to adversarial attacks [26, 43] (i.e., unexpected inputs), and resistance to distribution shifts [6, 19]. In this paper, we focus on robustness to general variations in environments within the context of RL. One approach is to introduce a robust MDP, which adds an uncertainty transition set to the MDP, and then use it to estimate the worst-case expected return among perturbed environments [7, 21, 22]. Another approach is domain randomization [27, 35, 36, 44]. Domain randomization involves randomizing the environment dynamics to train policies that can adapt to different conditions. For instance, [27] leverages domain randomization to develop an RL policy that is robust to variations in environment dynamics, thereby achieving robust sim-to-real transfer. The other approach is modeling environment dynamics via meta-learning [25] or supervised learning [19]. [19] introduces a separate context-aware dynamics model that infers transitions to adapt to dynamic changes by capturing the environment context. These methods have been applied to several robotics problems, including simulated Mujoco environments and manipulating robotic arms, but have not been considered for the IPP problem.

## 2.3 Gaussian Process Regression

In IPP, the environment of *interest* is typically modeled as a continuous function $\zeta : \mathcal{E} \to \mathbb{R}$ and is unknown. In this work, we consider a spatio-temporal environment $\mathcal{E} \subset \mathbb{R}^3$ (2D space + 1D time). Gaussian Processes (GP) are widely used to represent not only spatially correlated phenomena [4, 11, 30, 41], but also spatio-temporally changing environments [3, 13, 15]. The reason being, GPs can interpolate between discrete measurements and approximately represent the underlying phenomenon of interest: $\zeta \approx \mathcal{GP}(\mu, P)$. Given a set of $n'$ locations $\mathcal{X}^* \subset \mathcal{E}$ at which interest needs to be inferred, a set of $n$ observed locations $\mathcal{X} \subset \mathcal{E}$ and the corresponding measurements set $\mathcal{Y}$, the mean and covariance of the GP are regressed as: $\mu = \mu(\mathcal{X}^* + K(\mathcal{X}^*, \mathcal{X})[K(\mathcal{X}, \mathcal{X}) + \sigma_n^2 I]^{-1}(\mathcal{Y} - \mu(\mathcal{X}))$, $P = K(\mathcal{X}^*, \mathcal{X}^*) - K(\mathcal{X}^*, \mathcal{X})[K(\mathcal{X}, \mathcal{X}) + \sigma_n^2 I]^{-1} \times K(\mathcal{X}^*, \mathcal{X})^T$, where $K(\cdot)$ is a pre-defined kernel function, $\sigma_n^2$ is a parameter

representing the measurement noise, and $I$ is a $n \times n$ identity matrix. Our work uses the Matérn 3/2 kernel following [4, 10, 28, 39].

## 2.4 Informative Path Planning

The general IPP problem aims to find an optimal trajectory $\psi^*$ in the space of all available trajectories $\Psi$ to optimize an information-theoretic objective function:

$$\psi^* = \arg\max_{\psi \in \Psi} I(\psi), s.t.\ C(\psi) \leq B, \tag{2.1}$$

where $I : \psi \to \mathbb{R}^+$ represents the information gain from measurements obtained along the trajectory $\psi$, $C : \psi \to \mathbb{R}^+$ maps a trajectory $\psi$ to its execution cost, and $B$ denotes the robot's budget limit, e.g., path length, time, or energy. In this work, we consider path-length constraints and define the information gain as $I(\psi) = \mathrm{Tr}(P^-) - \mathrm{Tr}(P^+)$, where $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix, and $P^-$ and $P^+$ represent the prior and posterior covariances, respectively, obtained before and after taking measurements along the trajectory $\psi$, following prior works [2, 4, 28]. The information gain is computed from sensor measurements taken at fixed intervals along the trajectory $\psi$. Measurements are collected each time the robot travels a fixed distance, and as a result, the total number is determined by the path-length budget $B$.

Optimizing Eq. 2.1, i.e., solving IPP, is known to be NP-hard. Consequently, computationally tractable approximate methods, such as sampling techniques that explore a complex space by generating random samples of possible solutions, have been proposed [1, 11, 14, 17, 42]. However, these sampling-based methods still require heavy computation at test time, which restricts their use in real-world applications.

**RL-based IPP:** To address the aforementioned problem, RL has been utilized to learn an IPP [4, 5, 9, 31, 39]. Most RL-based IPP algorithms are composed of three modules: (a) creating a representation of the entire search map, (b) modeling environmental phenomena, and (c) training an RL agent. One representative example is CAtNIPP [4]. CAtNIPP trains an RL policy for IPP in static, time-invariant 2D environments. Before the training starts, CAtNIPP uses a *probabilistic roadmap* (PRM) [18] that covers the continuous search domain to decrease the complexity of the search space. PRM generates a route graph, $G = (V, E)$, where $V$ and $E$ are

sets of nodes and edges, respectively, and each node has $k$ neighbor nodes. Here, the agent is initialized on a randomly chosen node. Next, CAtNIPP leverages GP regression [33] to model the spatial phenomena of the search space, and the output of the GP regression is referred to as the belief of the phenomena. At each time step, the agent observes a measurement at the current node and then uses it to update the belief $\mathcal{GP}(\mu, P)$.

Lastly, for training the RL policy, the graph augmented by the updated belief—where each node $v_i' = (v_i, \mu(v_i), P(v_i)) \in V$—along with the planning state, which includes the current location, the budget, and the executed trajectory so far, is used as input for the RL policy. Based on such information, the agent chooses one of the neighboring nodes to move to (action). The reward function is based on the previously described information gain, which represents the reduction in uncertainty of GP regression, and is written as $r(t) = (Tr(P^{t-1}) - Tr(P^t))/Tr(P^{t-1})$. Summarized above, CAtNIPP trains an RL policy that chooses the neighboring node to move to based on the updated belief-augmented graph to maximize the expected sum of the reduction in uncertainty of GP regression. Additionally, CAtNIPP constructs the RL policy with an attention-based encoder and an LSTM-based [12] decoder module.

In addition to CAtNIPP, several RL-based IPP algorithms, including [39], where a dynamic graph is proposed to ensure collision-free navigation in 3D environments, have been introduced. The prior works have been shown to be effective in static, time-invariant environments however, none have considered spatio-temporal environments or variations in environmental dynamics, where the testing environments differ from those on which the agent was trained. To the best of our knowledge, our work is the first to propose an RL-based IPP policy for spatio-temporal environments and rigorously address its robustness.

# Chapter 3

# Methodology

## 3.1 Motivation

As described in Sec. 2.1, the standard RL framework assumes a fixed, stationary transition probability, $p(s'|s,a)$, which captures the environment dynamics. Thus, the optimal RL policy in an environment with a specific transition probability may be suboptimal in an environment with a different transition probability. In order to formalize this, we consider the distribution of MDPs, where the transition probability $p(s'|s,a,c)$ is conditioned on environment characteristics $c$. Here, the environment characteristics depend on the domain; e.g., in the wildfire domain, $c$ comprises several components that affect fire dynamics - including fuel/vegetation, and wind velocity - but are unknown to the agent.

The existing RL-based IPP algorithms [4, 39] are designed for static, time-invariant environments. Since this approach does not explicitly account for varying environment dynamics, it may lead to suboptimal performance when the environment dynamics deviate from those seen during training. As we will discuss in Sec. 4.3.1, an RL policy trained on specific environment characteristics often performs well only in the same environment, but struggles when deployed in environments with different characteristics. Since the agent can encounter various environments with different characteristics, it should be capable of handling these variations.

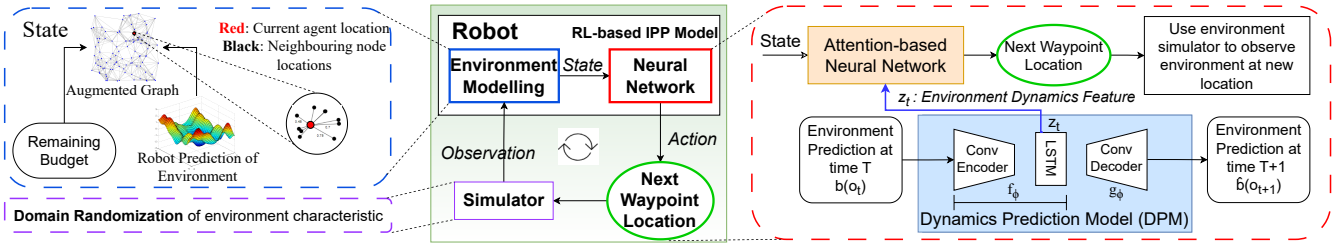


Figure 3.1: Overview of our approach: (Left) Environment modeling to build the input for the policy and domain randomization for the simulator. (Middle) The overall operation of the proposed IPP algorithm. (Right) Generating the next waypoint (action) using the RL policy and the proposed DPM (blue). DPM predicts the next environment state, and the environment dynamics feature is extracted from the hidden layer to serve as input to the RL policy.

## 3.2 Proposed Method: DyPNIPP

We aim to train an RL policy for IPP that performs well in different environments with varying dynamics—and is robust against the variations. To achieve this, we present DyPNIPP: an environment **Dy**namcs **P**rediction **N**etwork for **IPP**. The proposed framework comprises of two components: (1) domain randomization (DR), and (2) an additional network along with the RL policy designed to capture the environment dynamics by predicting the belief of the next observation, named dynamics prediction model. Note that DyPNIPP can be implemented on top of any RL-based IPP algorithm.

### 3.2.1 Domain Randomization

In order to allow the RL agent to encounter a diverse range of environment characteristics, we adopt domain randomization, which involves randomizing the environment dynamics by sampling characteristics from a specified range. This can be interpreted as constructing the environment dynamics by marginalizing the dynamics over the characteristics, written as $p(s'|s,a) = \mathbb{E}_c\left[p(s'|s,a,c)\right]$. That is, we sample a characteristic from a prior distribution of characteristics (e.g., a uniform distribution Uniform$[1,10]$ for the fuel/vegetation in the wildfire domain) for each episode and then train the RL policy. As we show in Sec. 4.3.1, this domain randomization alone is not enough to find the optimal policy across the characteristics. This is

because such a policy is optimal for an environment with marginalized transitions, $\mathbb{E}_c\left[p(s'|s,a,c)\right]$, implying it is optimal only for the averaged environment, and not for an individual environment with specific characteristics.

### 3.2.2 Dynamics Prediction Model

To address the aforementioned problem, we introduce a dynamics prediction model (DPM) capable of extracting features of environment dynamics. The proposed DPM uses an encoder consisting of convolutional layers and a LSTM layer, and a decoder involving a dense layer and transposed convolutional layers. The encoder and the decoder parameterized by $\phi_{enc}$ and $\phi_{dec}$ are given by $f(b(o_t), h_t; \phi_{enc})$ and $g(z_t; \phi_{dec})$ respectively. Here, $b(o_t)$ and $h_t$ are the belief of the observation and the hidden vector of the LSTM at time step $t$, respectively. $z_t$ is the output of the encoder referred to as the latent vector. We design the DPM to predict the belief of the next observation based on the belief of the current observation, which is given by GP regression. The loss function $\mathcal{L}_{DPM}$ is written as:

$$\mathcal{L}_{DPM}(\phi_{enc}, \phi_{dec}) = \mathbb{E}\left[\|y_t - g(z_t; \phi_{dec})\|^2\right] \tag{3.1}$$

where $z_t = f(b(o_t), h_t; \phi_{enc})$ and $y_t = b(o_{t+1})$. This learning process allows the hidden features of the DPM to represent the environment dynamics. The DPM plays a crucial role in enabling the agent to infer the underlying environment dynamics based on the agent's observations. We use the output of the DPM encoder, $z_t$, as an environment-context feature ($z_t$ is 16-dimensional feature).

### 3.2.3 RL policy

Based on the two proposed components, we train an RL policy that selects the next waypoint to move to. Note that DPM can be combined with any RL-based IPP policy. In this work, we use CAtNIPP, which is described in Sec. 2.4. The key difference is that the environment-context feature of DPM is injected into the RL policy, which is consequently represented as $\pi(a_t|s_t, z_t)$. The detailed architecture of the RL policy to generate action is as follows: The RL policy consists of an attention-based encoder and an attention and LSTM-based decoder. The encoder takes the

belief-augmented graph to generate a spatially aware embedding for all nodes. An additional node-embedding is generated using the planning state, which comprises of the remaining budget, executed trajectory, and the augmented graph. Here, we use our environment-context feature $z_t$ from the DPM module as an additional input with these current node-embeddings and pass them to the LSTM. A decoder is then used to select the most informative neighbor of the robot as an action. The action is selected through a cross-attention network. This attention mechanism computes attention weights by comparing the current node's features with the neighboring nodes' features. These weights reflect the relevance of each neighbor in the context of the current planning state, and also act as the policy. This process uses the budget, the executed trajectory, and the spatial as well as the environmental embeddings. Thus, the policy can generate actions which are not only feasible given the robot's operational budget, but are also informative and cognizant of the robot's current state and the environment dynamics

Note that the RL policy's LSTM focuses on capturing past information rather than dynamics since it is trained to maximize rewards. In contrast, the DPM's LSTM is explicitly trained to predict environment dynamics, making it better suited to capture variations in the underlying environment. To train this policy, we use PPO, following prior work [4].

Summarized above, the overall operation is as follows: (1) we initialize the environment with domain randomization, (2) the robot observes the environmental phenomena and updates the GP, (3) the GP-augmented graph and the remaining budget are fed into the network consisting of the RL policy and DPM, and (4) the environment dynamic feature from the DPM is used by the RL policy. (5) The RL policy generates an action that indicates the next waypoint location for the robot to move to. Repeating this procedure, we train the RL policy and DPM. This procedure is illustrated in Fig. 3.1.

# Chapter 4

# Experiments

## 4.1 Environment Setup

We evaluate the proposed framework on two domains: a wildfire domain given by the Fire Area Simulator (FARSITE) [8] model, and an air temperature prediction domain.

For the wildfire domain, we use FireCommander [34] simulator to generate spatio-temporal environments. The fire's growth rate (i.e. fire propagation velocity) is a function of fuel and vegetation coefficient ($F_c$), wind speed ($U_c$), and wind azimuth ($\theta_c$). The first-order firespot dynamics $\dot{q_t}$ are estimated for each propagating spot $q_t$, where $q$ is a 2-dimensional vector representing the X-Y coordinates, by $\dot{q_t} = C(F_c, U_c)\mathcal{D}(\theta_c)$, where $\mathcal{D}(\theta_c) = [\sin(\theta_c), \cos(\theta_c)]$. Here, $C(F_c, U_c)$ can be calculated as: $C(F_c, U_c) = F_c \left(1 - LB(U_c)/(LB(U_c) + \sqrt{GB(U_c)})\right)$, where $LB(U_c) = 0.936e^{0.256U_c} + 0.461e^{-0.154U_c} - 0.391$ and $GB(U_c) = LB(U_c)^2 - 1$. For evaluation in terms of robustness, we consider different environments with different fuel/vegetation coefficient $F_c$ while choosing the wind azimuth ($\theta_c$) randomly. We also conduct evaluations by varying the number of fire origins within the environment.

For the air temperature domain, we use real-world data [16] with monthly temperature readings from 1948 to 2022, discretized at 2.5° latitude × 2.5° longitude over a 360° × 180° area. We sample a small region for both training and testing (e.g., Fig 4.1). Note that unlike the wildfire domain where the environment was synthetically generated by varying known parameters across episodes, the air temperature

| Model used | Environment Parameter (Fuel and Vegetation coefficient) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | budget = 7 | | | budget = 11 | | | budget = 15 | | |
| | $F_c$=1 | $F_c$=5 | $F_c$=10 | $F_c$=1 | $F_c$=5 | $F_c$=10 | $F_c$=1 | $F_c$=5 | $F_c$=10 |
| Sampling-based IPP (non-RL) | 523.1 ± 592.7 | 551.9 ± 505.5 | 550.2 ± 469.7 | 949.4 ± 1161.2 | 932.0 ± 870.2 | 997.4 ± 916.6 | 1305.2 ± 1643.1 | 1372.7 ± 1289.7 | 1367.1 ± 1142.4 |
| CAtNIPP (trained with $F_c = 1$) | 460.5 ± 37.1 | 470.8 ± 38.3 | 485.6 ± 38.6 | 450.4 ± 40.9 | 465.2 ± 38.6 | 483.7 ± 41.1 | 442.0 ± 37.9 | 464.1 ± 42.7 | 479.3 ± 45.5 |
| CAtNIPP (trained with $F_c = 5$) | 428.2 ± 28.5 | 427.0 ± 31.8 | 429.2 ± 34.4 | 367.2 ± 27.7 | 372.9 ± 32.2 | 380.2 ± 37.4 | 352.8 ± 26.7 | 367.9 ± 30.4 | 375.6 ± 36.4 |
| CAtNIPP (trained with $F_c = 10$) | 496.4 ± 51.4 | 498.8 ± 52.2 | 498.6 ± 50.0 | 464.8 ± 51.9 | 471.3 ± 55.5 | 481.7 ± 53.8 | 457.8 ± 53.2 | 469.6 ± 54.9 | 473.5 ± 50.9 |
| CAtNIPP + DR | 419.3 ± 30.3 | 422.8 ± 29.3 | 427.5 ± 32.6 | 378.7 ± 25.9 | 387.1 ± 35.9 | 392.7 ± 37.0 | 376.4 ± 28.7 | 389.3 ± 35.3 | 393.1 ± 37.0 |
| `DyPNIPP` **(Ours)** | **401.3 ± 25.5** | **403.5 ± 26.4** | **403.2 ± 27.3** | **349.1 ± 25.3** | **349.7 ± 24.9** | **348.9 ± 26.5** | **340.9 ± 23.0** | **342.8 ± 27.1** | **347.3 ± 28.7** |

Table 4.1: Covariance Trace comparison across 200 instances for three budgets; lower values indicate better performance.

| Model used | Environment Parameter (Fuel and Vegetation coefficient) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | budget = 7 | | | budget = 11 | | | budget = 15 | | |
| | $F_c =$ 1 | $F_c =$ 5 | $F_c =$ 10 | $F_c =$ 1 | $F_c =$ 5 | $F_c =$ 10 | $F_c =$ 1 | $F_c =$ 5 | $F_c =$ 10 |
| Sampling-based IPP (non-RL) | 7.2 ± 6.8 | 9.5 ± 6.5 | 10.9 ± 7.9 | 12.5 ± 14.1 | 14.2 ± 10.9 | 16.0 ± 12.8 | 14.1 ± 14.8 | 16.8 ± 13.6 | 17.9 ± 14.2 |
| CAtNIPP (trained with $F_c = 1$) | 6.2 ± 1.3 | 13.4 ± 3.1 | 9.5 ± 2.2 | 9.5 ± 2.0 | 13.2 ± 2.4 | 16.1 ± 4.1 | 13.7 ± 2.6 | 21.8 ± 4.8 | 25.0 ± 5.3 |
| CAtNIPP (trained with $F_c = 5$) | 6.3 ± 1.3 | 8.2 ± 1.6 | 9.0 ± 2.1 | 9.4 ± 1.5 | 12.1 ± 2.1 | 13.6 ± 2.7 | 15.0 ± 2.7 | 18.7 ± 2.8 | 18.7 ± 2.8 |
| CAtNIPP (trained with $F_c = 10$) | 6.2 ± 1.5 | 8.4 ± 2.1 | 9.7 ± 2.6 | 10.2 ± 2.7 | 13.3 ± 3.4 | 15.4 ± 4.1 | 16.2 ± 3.6 | 21.7 ± 4.7 | 20.9 ± 3.0 |
| CAtNIPP + DR | 6.1 ± 1.3 | 8.2 ± 1.6 | 9.9 ± 2.0 | 9.1 ± 1.7 | 12.9 ± 2.7 | 15.3 ± 3.5 | 14.9 ± 2.4 | 21.4 ± 4.6 | 24.0 ± 5.6 |
| `DyPNIPP` **(Ours)** | **5.2 ± 1.0** | **6.9 ± 1.4** | **7.8 ± 1.5** | **7.7 ± 1.3** | **10.9 ± 1.9** | **11.5 ± 2.3** | **10.3 ± 1.6** | **14.0 ± 2.6** | **14.7 ± 2.9** |

Table 4.2: Cumulative RMSE comparison across 200 instances for three budgets; lower values indicate better performance.

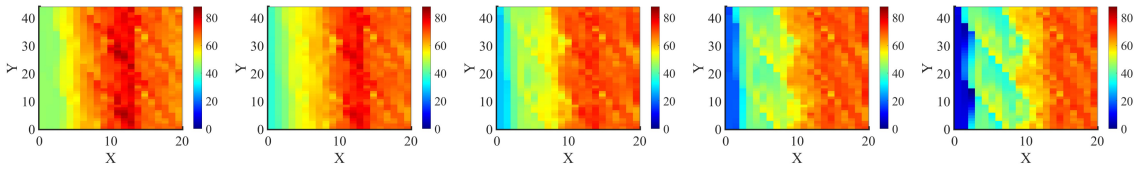

Figure 4.1: Example of air temperature progression over a fix-sized 112.5 degree latitude ($Y$) by 50 degree longitude ($X$) area.

naturally varies within each episode with unknown and evolving dynamics. Through our experiments, we show that `DyPNIPP` can handle both the types of environments.

### 4.1.1 Performance Comparison in Air Temperature Prediction

While we synthetically vary environment dynamics in the wildfire domain, we also compare `DyPNIPP` with CAtNIPP on a real-world dataset that naturally includes environment variations within an episode. As seen in Table 4.3, `DyPNIPP` outperforms `CAtNIPP` in terms of the covariance trace on the real dataset.

| Model used | Covariance Trace | | |
|---|---|---|---|
| | budget=4 | budget=7 | budget=10 |
| CAtNIPP | 238.7 ± 42.9 | 133.3 ± 32.9 | 109.8 ± 28.8 |
| `DyPNIPP` **(Ours)** | **215.7 ± 44.1** | **126.35 ± 21.8** | **102.7 ± 15.4** |

Table 4.3: Covariance Trace comparison in air temperature domain; lower values indicate better performance.

## 4.2 Training Details

In the FireCommander simulator, both higher $F_c$ and $U_c$ result in a higher fire spread rate ($F_c, U_c > 0$). Since a higher $F_c$ implies highly flammable fuel, it also means that the fuel can get exhausted more quickly, resulting in a quicker decay of fire. Although theoretically both coefficients can be any positive numbers, 10 is given as the recommended limit for both. Note that the fuel and vegetation coefficient $F_c$ is the most dominant factor affecting the fire spread. Thus, for all our

| Model used | Environment Parameter (Fuel and Vegetation coefficient) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | n(fires) = 1 | | | n(fires) = 3 | | | n(fires) = 5 | | |
| | $F_c$=1 | $F_c$=5 | $F_c$=10 | $F_c$=1 | $F_c$=5 | $F_c$=10 | $F_c$=1 | $F_c$=5 | $F_c$=10 |
| CAtNIPP + DR | 10.3 ± 2.3 | 14.5 ± 3.0 | 16.5 ± 3.4 | 13.9 ± 2.6 | 16.5 ± 3.6 | 19.7 ± 2.9 | 14.0 ± 2.6 | 18.3 ± 3.1 | 20.5 ± 3.7 |
| DyPNIPP **(Ours)** | **9.3 ± 2.0** | **11.9 ± 3.1** | **13.0 ± 3.3** | **12.2 ± 2.6** | **14.4 ± 3.6** | **15.0 ± 3.8** | **13.4 ± 2.7** | **15.4 ± 3.4** | **15.5 ± 3.2** |

Table 4.4: Cumulative RMSE comparison with respect to the number of fires. Policies were trained in environments with up to 3 fires. DyPNIPP shows robustness, maintaining low RMSE even in out-of-distribution environments 5-fire origins.

experiments, we fix the wind speed $U_c$ (weaker factor) at 5 and vary $F_c$ (dominant factor) to train and test for robustness. The wind azimuth ($\theta_c$) is chosen randomly for each episode. We use domain randomization to make our policy robust to changes in these parameters which impact the fire spread dynamics. While training our policy, we randomly choose a $F_c$ in between $[1, 10]$ for every episode (unless otherwise stated). Based on these sampled environment parameters, we use FireCommander to generate the spatio-temporal wildfire environment and normalize the field dimensions to construct the true interest map of a unit square $[0, 1]^2$ size. Since there are no initial observations of the environment, the robot's belief starts as a uniform distribution $\mathcal{GP}(0, P^0), P^0_{i,i} = 1$. The start and destination positions are randomly generated in $[0, 1]^2$. We train our policy with a fixed number (200) of nodes for our graph, where the number of neighboring nodes is fixed to $k = 20$, and the budget is randomized between $[7, 9]$. Note that the graph is reinitialized for each episode. A measurement is obtained every time the robot has traveled 0.2 units from the previous measurement. We set the max episode length to 256 time steps, and the batch size to 32. We use the Adam optimizer with learning rate $10^{-4}$, which decays every 32 steps by a factor of 0.96.

`DyPNIPP` trains a RL agent on top of CAtNIPP [4] utilizing PPO [32] for the training. For each training episode, PPO runs 8 iterations. Our model is trained on a workstation using AMD EPYC 7713 CPU, and a single NVIDIA RTX 6000 Ada Gen GPU. We use Ray [24] to distribute the training process and run 16 IPP instances in parallel, and require around 2 hours to converge.

| Prediction | Environment Parameter | | | | | |
|---|---|---|---|---|---|---|
| | budget=7 | | | budget=11 | | |
| | $F_c$=1 | $F_c$=5 | $F_c$=10 | $F_c$=1 | $F_c$=5 | $F_c$=10 |
| $b(o_t)$ | $7.2 \pm 1.2$ | $8.9 \pm 1.5$ | $9.9 \pm 1.5$ | $10.8 \pm 1.6$ | $13.7 \pm 1.9$ | $14.4 \pm 2.1$ |
| $b(o_{t+1}) - b(o_t)$ | $5.3 \pm 1.1$ | $6.8 \pm 1.4$ | $7.9 \pm 1.5$ | $8.0 \pm 1.4$ | $10.7 \pm 1.9$ | $11.7 \pm 2.3$ |
| $b(o_{t+1})$ (**Ours**) | $5.2 \pm 1.0$ | $7.0 \pm 1.4$ | $7.8 \pm 1.5$ | $7.8 \pm 1.3$ | $10.9 \pm 1.9$ | $11.5 \pm 2.3$ |

Table 4.5: Ablation on dynamics prediction model design.

## 4.3 Experimental Results

In this section, we examine (1) whether `DyPNIPP` enhances the robustness against environment variations such as the fuel coefficient $F_c$ and number of fires, (2) the effectiveness of our DPM design, and (3) how the environment-context latent feature varies with respect to the dynamics.

### 4.3.1 Performance Comparison of Variation in Fuel Coefficient

We include three main baselines: First, we include three CAtNIPP policies trained on a fixed $F_c \in \{1, 5, 10\}$. Second, we include CAtNIPP combined with domain randomization, where $F_c$ is randomly sampled between $[0, 10]$ for every episode. We refer to this method as CAtNIPP + DR in Table 4.2. Since CAtNIPP was designed for static, time-invariant environments, we adopt it to spatio-temporal environments by adding time to the set of regressor variables in the GP. Lastly, we evaluate a sampling-based informative planner, where the target location is selected by maximizing an objective that balances predictive entropy at the target location and the travel distance from the robot's current position. Specifically, the objective is the predictive entropy minus the distance cost. To initialize the spatio-temporal GP for this method, we provide 100 initial observations sampled at random locations.

**Performance Metrics:** We compare `DyPNIPP` with these baselines across three environments, each characterized by a different $F_c \in \{1, 5, 10\}$, and three different budgets ($B \in 7, 11, 15$). We use two metrics for evaluation: the covariance trace $Tr(P)$ and the cumulative root mean squared error (RMSE). The covariance trace

provides information about the confidence of the robot's GP model—representing the uncertainty remaining at the end of the mission, which aligns with the IPP objective described in section 2.4. Since this does not necessarily correlate with the accuracy of the environment prediction, we introduce the RMSE between the environment prediction and the actual ground truth after every step of the agent's trajectory. We provide the results in terms of the metrics in Table 4.1 and 4.2. Based on the results, we observe the following:

• **The prior RL-based IPP algorithm lacks robustness**: CAtNIPP trained on a specific $F_c$ exhibits suboptimal performance in environments with a different $\bar{F}_c$. For example, in the case of a budget of 15, CAtNIPP trained on $F_c = 1$ outperforms CAtNIPP trained on $F_c = 5$ and $F_c = 10$ in an environment with $F_c = 1$, whereas it performs poorly in environments with $F_c = 5$ and $F_c = 10$.

• **Domain randomization alone is not sufficient**: CAtNIPP+DR performs intermediate to the CAtNIPP policies trained on $F_c = 1, 5, 10$. This implies that `DyPNIPP` without environment dynamic prediction converges to the optimal policy for averaged environment dynamics, which is suboptimal for each individual environment dynamic.

• `DyPNIPP` **is robust against the variation in environment dynamics**: `DyPNIPP` outperforms the baselines on both metrics. In addition, `DyPNIPP` shows similar covariance trace performance regardless of the underlying fuel distribution. These are strong pieces of evidence supporting the **robustness** of our approach against variations in environment dynamics.

### 4.3.2 Performance Comparison of Variation in number of fires

We additionally study the effect on our policy when tested in environments with different number of fires, each randomly originating at different timestamps within an episode. We evaluate `DyPNIPP` and CAtNIPP + DR, both trained in an environment with up to 3 fires, on environments with 1, 3, and 5 fires each, in terms of cumulative RMSE. Note that the 5-fire environment is out-of-distribution for both the policies. As seen in Table 4.4, our proposed method outperforms CAtNIPP + DR in all considered settings, implying that `DyPNIPP` handles variations in the number of fires

as well as variations in the fuel coefficient.

### 4.3.3 Performance Comparison in Air Temperature Prediction

While we synthetically vary environment dynamics in the wildfire domain, we also compare `DyPNIPP` with CAtNIPP on a real-world dataset that naturally includes environment variations within an episode. As seen in Table 4.3, `DyPNIPP` outperforms `CAtNIPP` in terms of the covariance trace on the real dataset.

## 4.4 Analysis: Dynamics Prediction Model

We proposed to design the DPM to predict the belief of the next observation to capture the environment dynamics. To verify this, we conduct an ablation study comparing two modified versions of `DyPNIPP` that predict (1) the belief of the current observation and (2) the belief difference between the current observation and the next observation. The corresponding result is shown in Table 4.5. We can see that predicting the belief of the current observation performs worse than `DyPNIPP` and the version of `DyPNIPP` that predicts the belief difference, both of which involve predicting the next observation. This is because predicting the belief of the current observation is not capable of capturing the environment dynamics. We observe that the version of `DyPNIPP` predicting the belief difference also performs well, even though its performance is slightly lower than the original `DyPNIPP`.

Additionally, to verify that the DPM implicitly learns the environment dynamics, we investigate the latent embeddings generated by the DPM's LSTM using t-SNE [38]. We collect the final 16-dimensional embeddings at the end of the episode for three fuel/vegetation coefficients, $F_c \in \{1, 5, 10\}$, with 200 seeds each. These embeddings are visualized in a 2-dimensional space using t-SNE. As shown in Fig. 4.2, we observe a clear progression of the latent embeddings as the fuel/vegetation coefficient increases from 1 (purple) to 5 (green), and then to 10 (yellow).

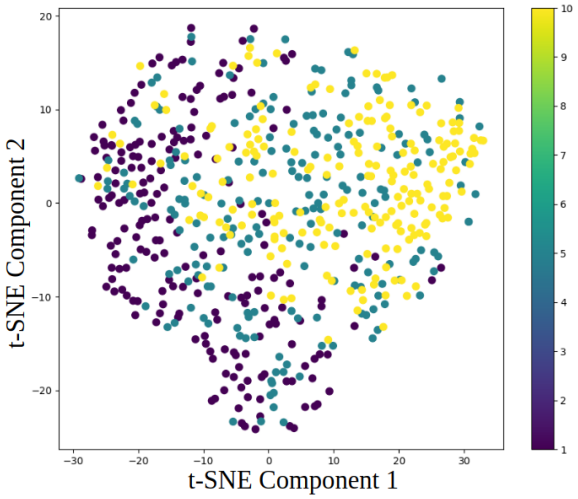


Figure 4.2: t-SNE plot of environment-context feature. Fuel/vegetation coefficient: 1 (purple), 5 (green), 10 (yellow). Higher coefficient means faster spread rate.

## 4.5 Analysis: Planning Time

The benefit of using RL for the IPP problem includes both improved performance and decreased computation time. Using a graph size of 200 with a budget 15 units, `DyPNIPP`, CAtNIPP, and the sampling-based non-RL IPP method take $2.60 \pm 0.26$, $1.92 \pm 0.29$, and $18.05 \pm 12.90$ seconds respectively to plan the entire path. `DyPNIPP` takes slightly more time compared to CAtNIPP due to the introduction of an additional network on top of CAtNIPP; however, `DyPNIPP` is much faster than the sampling-based methods.

## 4.6 Experimental Validation on Real Robot

We provide experimental validation on a real robot to verify that our IPP model, trained in a simulator, can be applied in the real world. For this, we project the spatio-temporal wildfire environment onto a physical $1.5\ m \times 1.5\ m$ arena, maintaining consistency in configuration between simulation and physical deployment. We used the Khepera-IV robot, equipped with a Raspberry Pi 3 and a camera module. 4.3 In this experiment, we first trained a policy in simulation and then tested it using the Khepera-IV robot and the arena. Here, the robot can only observe the intensity of

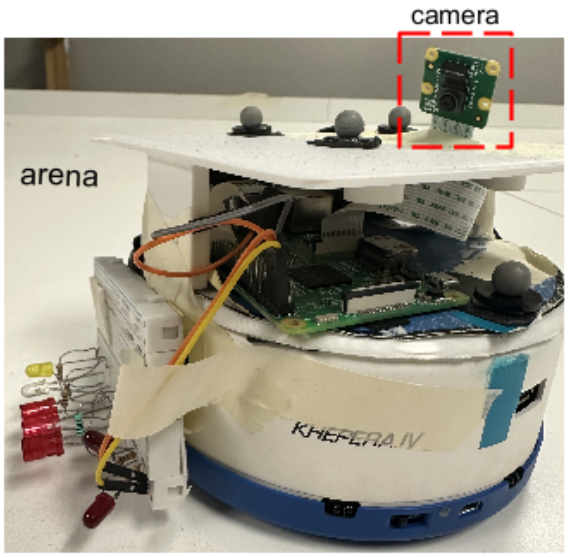


Figure 4.3: Khepera-IV robot used for the real world experiments. Raspberry Pi 2 and the camera module is attached onboard to assist with the sensing.

the fire, which is an environmental phenomenon, at its current location. It updates its belief via GP regression using the observation, and the updated belief is used as input for the next forward pass of our policy. We use a budget of 12 m for this experiment, which corresponds to 8 units in a unit grid. It is observed that the robot trained in the simulator successfully finds a path that maximizes the information gain, as shown in Fig. 4.4. Through this experiment, we also demonstrate the fast decision-making time of our policy (less than 0.25 s on an Intel Xeon CPU). This experiment shows that `DyPNIPP` can be deployed on a robot in the spatio-temporal environment. The full video can be accessed here.

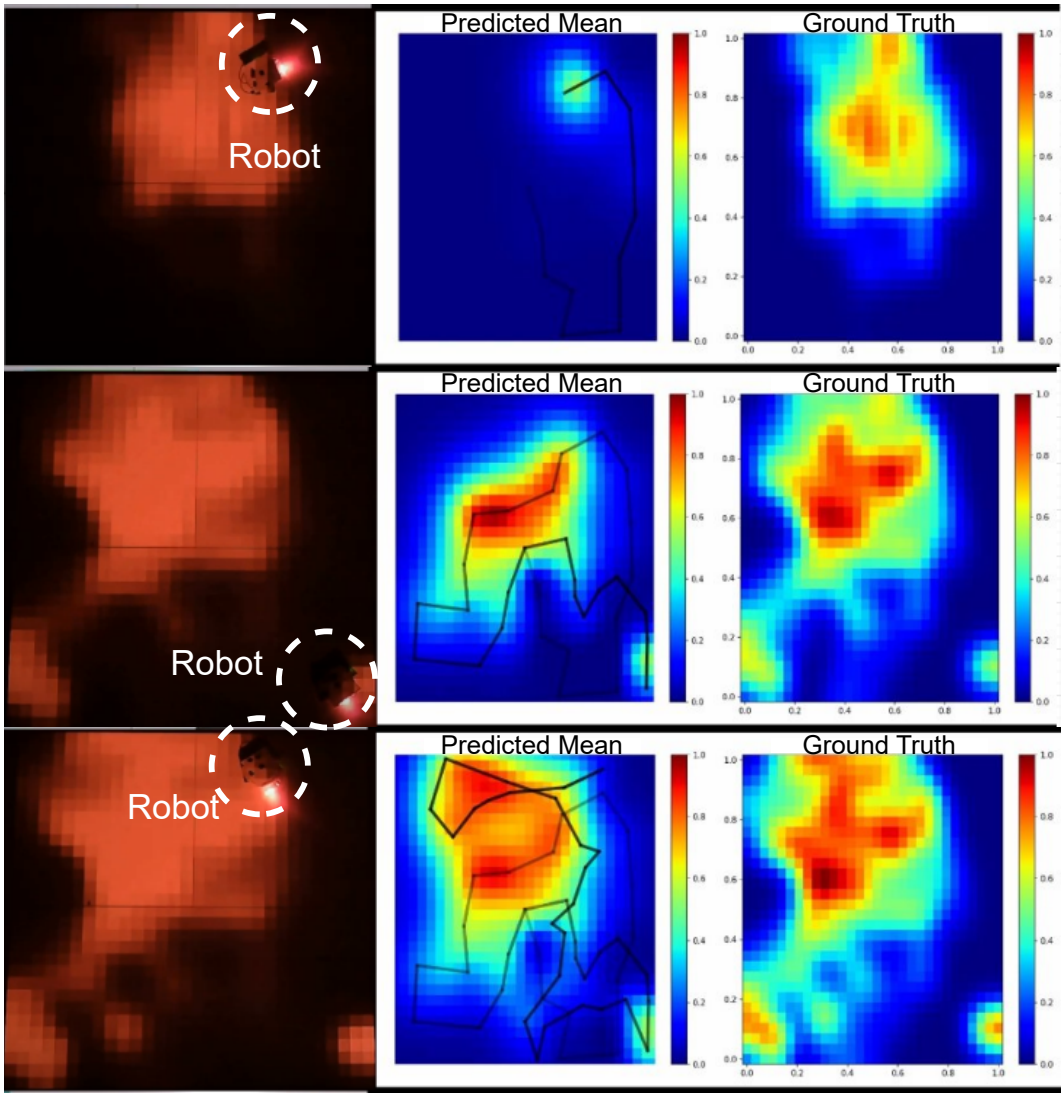


Figure 4.4: Experimental validation of `DyPNIPP` on the Khepara-IV robot. Left: robot performing informative path planning, observing environmental phenomena at its location, with red areas indicating higher values. The right figures display the predicted phenomena and ground truth environment, with time progressing from the first to the third row. Predictions improves as the robot explores more areas.

# Chapter 5

# Conclusions

In this paper, we propose DyPNIPP, an RL-based framework for IPP that not only works in spatio-temporal environments but is also robust against variations in environmental dynamics. DyPNIPP introduces two components on top of an RL-based IPP algorithm. First, DyPNIPP includes domain randomization during training to encourage the agent to encounter variations in environment dynamics. Second, DyPNIPP includes a dynamics prediction model that predicts the belief of the next observation, allowing the agent to figure out the current environmental dynamics. We show that DyPNIPP is superior to existing IPP methods in terms of solution quality across diverse environmental dynamics, demonstrating its greater robustness. In addition, we provide a physical robot experiment to showcase real-time planning, highlighting its potential for real-life robotic applications.

**Limitations** Since we use a predefined graph for traversing, two limitations arise: (1) due to uniform sampling, some areas of interest may be unreachable because of insufficient graph coverage, and (2) the predefined graph cannot adapt to unknown obstacles. We expect DyPNIPP to work with dynamically sampled graphs to address obstacle avoidance [39]. Another limitation is that the policy's performance significantly depends on the robot's belief, which relies on GP hyperparameters. Different environment dynamics could have different "optimal" hyperparameters; however, this mapping is unknown.

**Future Work** Several extensions would enhance the applicability of this research to actual wildfire environments. First, real fires are accompanied by smoke, which

causes occlusions for visual sensors. Therefore, it becomes essential to predict smoke dynamics to guide the robot through areas with minimal smoke exposure. This remains an optimization problem as the robot must eventually reach its goal while avoiding smoke as much as possible. Such planning has important downstream applications: obtaining clear images of the underlying scene can enable 3D reconstruction, which would be valuable for front-line workers fighting the fire. Second, the proposed DPM could be integrated with a mixture of experts (MoE) framework, where multiple expert planners are specialized for different types of environment dynamics, and the DPM selects the appropriate expert for current conditions. Finally, future work will investigate how to integrate GP hyperparameter optimization into the existing learning framework.

# Appendix A

# Appendix

## A.1 Effect of Training with Different Budgets

We tried training our policy with two different budgets: (1) $B \in (7, 9)$, and (2) $B \in (30, 35)$. As we see in Fig. A.1 and A.2, the choice of budget impacts the policy performance to a huge extent. When trained with a lower budget, the policy learns to plan an informative path with much better exploratory behavior as compared to the other case. Hence, we stick to training policies with a lower budget and test them with different budget levels.

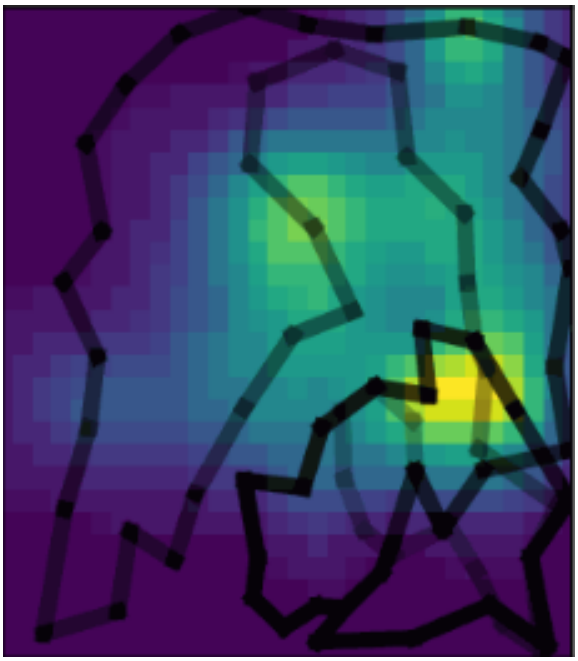

Figure A.1: Trained with budget $\in (7, 9)$

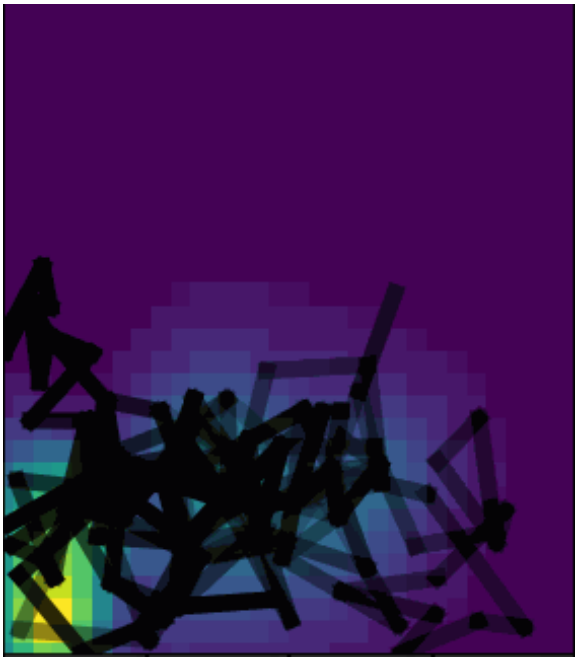

Figure A.2: Trained with budget $\in (30, 35)$

# Bibliography


[1] Sankalp Arora and Sebastian Scherer. Randomized algorithm for informative path planning with budget constraints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4997–5004. IEEE, 2017. 1, 2.4

[2] Jonathan Binney and Gaurav S. Sukhatme. Branch and bound for informative path planning. In *2012 IEEE International Conference on Robotics and Automation*, pages 2147–2154, 2012. doi: 10.1109/ICRA.2012.6224902. 2.4

[3] Jonathan Binney, Andreas Krause, and Gaurav S. Sukhatme. Optimizing waypoints for monitoring spatiotemporal phenomena. *The International Journal of Robotics Research*, 32(8):873–888, 2013. doi: 10.1177/0278364913488427. URL https://doi.org/10.1177/0278364913488427. 2.3

[4] Yuhong Cao, Yizhuo Wang, Apoorva Vashisth, Haolin Fan, and Guillaume Adrien Sartoretti. Catnipp: Context-aware attention-based network for informative path planning. In *Conference on Robot Learning*, pages 1928–1937. PMLR, 2023. 1, 1, 2.3, 2.4, 3.1, 3.2.3, 4.2

[5] Yuhong Cao, Rui Zhao, Yizhuo Wang, Bairan Xiang, and Guillaume Sartoretti. Deep reinforcement learning-based large-scale robot exploration. *IEEE Robotics and Automation Letters*, 2024. 1, 2.4

[6] Jongseong Chae, Seungyul Han, Whiyoung Jung, Myungsik Cho, Sungho Choi, and Youngchul Sung. Robust imitation learning against variations in environment dynamics. In *International Conference on Machine Learning*, pages 2828–2852. PMLR, 2022. 2.2

[7] Esther Derman, Daniel J Mankowitz, Timothy A Mann, and Shie Mannor. Soft-robust actor-critic policy-gradient. *arXiv preprint arXiv:1803.04848*, 2018. 2.2

[8] M. A. Finney. Farsite: Fire area simulator-model development and evaluation. *Res. Pap. RMRS-RP-4, Revised 2004. Ogden, UT: US Department of Agriculture, Forest Service, Rocky Mountain Research Station. 47 p.*, vol. 4, 1998. 4.1

[9] Srikar Babu Gadipudi, Srujan Deolasee, Siva Kailas, Wenhao Luo, Katia Sycara, and Woojun Kim. Offripp: Offline rl-based informative path planning. *arXiv*

*preprint arXiv:2409.16830*, 2024. 2.4

[10] Maani Ghaffari Jadidi, Jaime Valls Miro, and Gamini Dissanayake. Sampling-based incremental information gathering with applications to robotic exploration and environmental monitoring. *The International Journal of Robotics Research*, 38(6):658–685, 2019. 2.3

[11] Gregory Hitz, Enric Galceran, Marie-Ève Garneau, François Pomerleau, and Roland Siegwart. Adaptive continuous-space informative path planning for online environmental monitoring. *Journal of Field Robotics*, 34(8):1427–1449, 2017. 1, 2.3, 2.4

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. 2.4

[13] Kalvik Jakkala and Srinivas Akella. Multi-robot informative path planning from regression with sparse gaussian processes. *arXiv preprint arXiv:2309.07050*, 2023. 1, 2.3

[14] Austin Jones, Mac Schwager, and Calin Belta. A receding horizon algorithm for informative path planning with temporal logic constraints. In *2013 IEEE International Conference on Robotics and Automation*, pages 5019–5024. IEEE, 2013. 2.4

[15] Siva Kailas, Wenhao Luo, and Katia Sycara. Multi-robot adaptive sampling for supervised spatiotemporal forecasting. In *EPIA Conference on Artificial Intelligence*, pages 349–361. Springer, 2023. 1, 2.3

[16] Eugenia Kalnay, Masao Kanamitsu, Robert Kistler, William Collins, Dennis Deaven, Lev Gandin, Mark Iredell, Suranjana Saha, Glenn White, John Woollen, et al. The ncep/ncar 40-year reanalysis project. In *Renewable energy*, pages Vol1_146–Vol1_194. Routledge, 2018. 4.1

[17] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011. 1, 2.4

[18] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996. doi: 10.1109/70.508439. 2.4

[19] Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *International Conference on Machine Learning*, pages 5757–5766. PMLR, 2020. 2.2

[20] Jingsong Liang, Zhichen Wang, Yuhong Cao, Jimmy Chiun, Mengqi Zhang,

and Guillaume Adrien Sartoretti. Context-aware deep reinforcement learning for autonomous robotic navigation in unknown area. In *Conference on Robot Learning*, pages 1425–1436. PMLR, 2023. 1

[21] Daniel Mankowitz, Timothy Mann, Pierre-Luc Bacon, Doina Precup, and Shie Mannor. Learning robust options. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 2.2

[22] Daniel J Mankowitz, Nir Levine, Rae Jeong, Yuanyuan Shi, Jackie Kay, Abbas Abdolmaleki, Jost Tobias Springenberg, Timothy Mann, Todd Hester, and Martin Riedmiller. Robust reinforcement learning for continuous control with model misspecification. *arXiv preprint arXiv:1906.07516*, 2019. 2.2

[23] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *The International Journal of Robotics Research*, 43(4):572–587, 2024. 2.1

[24] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging ai applications, 2018. 4.2

[25] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018. 2.2

[26] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632*, 2017. 2.2

[27] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018. 2.2

[28] Marija Popović, Teresa Vidal-Calleja, Gregory Hitz, Jen Jen Chung, Inkyu Sa, Roland Siegwart, and Juan Nieto. An informative path planning framework for uav-based terrain monitoring. *Autonomous Robots*, 44(6):889–911, 2020. 2.3, 2.4

[29] Marija Popovic, Joshua Ott, Julius Rückin, and Mykel J Kochendorfer. Robotic learning for adaptive informative path planning. *arXiv preprint arXiv:2404.06940*, 2024. 1

[30] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005. ISBN 9780262256834. doi: 10.7551/mitpress/3206.001.0001. URL https://doi.org/10.7551/mitpress/3206.001.0001. 2.3

[31] Julius Rückin, Liren Jin, and Marija Popović. Adaptive informative path planning using deep reinforcement learning for uav-based active sensing. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4473–4479. IEEE, 2022. 1, 2.4

[32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. 2.1, 4.2

[33] Matthias Seeger. Gaussian processes for machine learning. *International journal of neural systems*, 14(02):69–106, 2004. 2.4

[34] Esmaeil Seraj, Xiyang Wu, and Matthew Gombolay. Firecommander: An interactive, probabilistic multi-agent environment for heterogeneous robot teams. *arXiv preprint arXiv:2011.00165*, 2020. (document), 1.1, 4.1

[35] Reda Bahi Slaoui, William R Clements, Jakob N Foerster, and Sébastien Toth. Robust visual domain randomization for reinforcement learning. *arXiv preprint arXiv:1910.10537*, 2019. 2.2

[36] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018. 2.2

[37] Gabriele Tiboni, Pascal Klink, Jan Peters, Tatiana Tommasi, Carlo D'Eramo, and Georgia Chalvatzaki. Domain randomization via entropy maximization, 2023. 1

[38] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9(nov):2579–2605, 2008. ISSN 1532-4435. Pagination: 27. 4.4

[39] Apoorva Vashisth, Julius Rückin, Federico Magistri, Cyrill Stachniss, and Marija Popović. Deep reinforcement learning with dynamic graphs for adaptive informative path planning. *arXiv preprint arXiv:2402.04894*, 2024. 1, 2.3, 2.4, 3.1, 5

[40] Yizhuo Wang, Yutong Wang, Yuhong Cao, and Guillaume Sartoretti. Spatio-temporal attention network for persistent monitoring of multiple mobile targets. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3903–3910. IEEE, 2023. 1

[41] Yongyong Wei and Rong Zheng. Informative path planning for mobile sensing with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 864–873. IEEE, 2020. 2.3

[42] Soo-Hyun Yoo, Andrew Stuntz, Yawei Zhang, Robert Rothschild, Geoffrey A Hollinger, and Ryan N Smith. Experimental analysis of receding horizon planning algorithms for marine monitoring. In *Field and service robotics*, pages 31–44.

Springer, 2016. 2.4

[43] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. *Advances in Neural Information Processing Systems*, 33:21024–21037, 2020. 2.2

[44] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, pages 737–744. IEEE, 2020. 2.2