Safe Planning and Exploration for Mobile Robots in Unknown, Hazardous Environments

Matthew Budd

Pembroke College

University of Oxford

Submitted in partial fulfilment of the requirements
for the Master of Engineering in Engineering Science
at the University of Oxford

Trinity Term 2020

Acknowledgements

This project benefited hugely from the help and support of a number of people within and without the Oxford Robotics Institute.

In particular I would like to thank Dr Bruno Lacerda and Dr Paul Duckworth for their constant input, feedback and insight into problems in their areas of expertise, and to Professor Nick Hawes for his consistent support with the project. Their help has been essential due to the variety of research topics covered in the text below, and I have been very fortunate to have had access to their knowledge.

I would also like to express my gratitude to Dr Andrew West at the University of Manchester for producing the gamma radiation environment data sets and maps, as well as for his insights into radiation physics.

Abstract

In this project, we tackle the problem of an autonomous mobile robot exploring an environment over which there is an unknown distribution of a hazard that may harm it - the exploration task is to safely gain as much knowledge of this distribution as possible. In particular, we make use of techniques from the fields of probabilistic planning under uncertainty and Gaussian process modelling of environmental features that affect robot safety.

A new model is proposed and developed for these safety-constrained exploration problems - an *Estimated MDP* - that integrates Gaussian process predictions within a Markov decision process structure, enabling the robot to better reason about the safety of its plans and also allow for efficient exploration. We analyse how the model relates to other formalisms designed to work with imperfect information, and use it to design a novel exploration algorithm that can handle more complex exploration problems than previous literature.

The algorithm is evaluated on simulated and real-world environments, including a gamma radiation dataset where the agent must explore an area with an *a priori* unknown distribution of radiation. The exploration system is then integrated into a mobile robot platform to demonstrate how it would function as the core of an autonomous exploration machine that ensures its own safety while gathering useful information on the distribution of hazards in its environment.

Contents

1 Introduction **1**

1.1 Background and Project Aims 1

1.2 Literature Review 2

1.3 Structure of this Report 4

2 Theoretical Approach **5**

2.1 Preliminaries 5

2.1.1 Markov Decision Processes 5

2.1.2 Constrained Reachability Optimisation and Linear Temporal Logic 6

2.1.3 Gaussian Process Regression 8

2.2 Additional Background Theory 12

2.2.1 Kullback–Leibler Divergence Between Two Multivariate Normal Distributions 12

2.2.2 Poisson Statistics of Radiation Exposure 12

2.3 Exploration Model 13

2.3.1 Reachability and Returnability in an MDP 14

2.3.2 Formal Problem Definition 15

2.3.3 Mapping from Continuous GP Domain to Discrete MDP States 15

2.4 Top-Level Exploration Algorithm for Probabilistic Transition Models 16

2.4.1 U-MDP Exploration Algorithm 16

2.4.2 Policy Safety Check 18

2.4.3 Probabilistic Determination of Reachability and Returnability 19

2.5 Confidence Bounds Approach to Path Planning 19

2.5.1 Confidence Bound Goal and Policy Generation Algorithm 20

2.6 Estimated MDP Approach to Path Planning 22

2.6.1 Estimated MDP Structure 22

2.6.2 Estimated MDP Goal and Policy Generation Algorithm 23

2.6.3 Goal Scoring Function 25

2.7 Extension of Estimated MDP Approach to Multi-Step GP Propagation 26

3 Implementation and Experiments 27

3.1 Software Architecture 27

3.2 Evaluation in Simulation 28

3.2.1 Programmatically Generated Water Depth Model 28

3.2.2 Single Radiation Source Dataset 30

3.2.3 Multiple Radiation Source Map 32

3.2.4 Expander State Determination 37

3.3 Further Discussion 38

3.3.1 Implementation Performance Evaluation 38

3.3.2 Modelling of Radiation Environments with Gaussian Processes 39

3.3.3 Additional Gaussian Process Model Considerations 41

4 System Integration with ROS 42

4.1 Software Architecture 42

4.1.1 Topological Navigation Stack 43

4.1.2 Safe Exploration Package 43

4.1.3 Radiation Tools Package 44

4.1.4 Output Visualisation Package 44

4.2 System Validation Outputs 45

5 Conclusions 46

5.1 Summary of Contributions 46

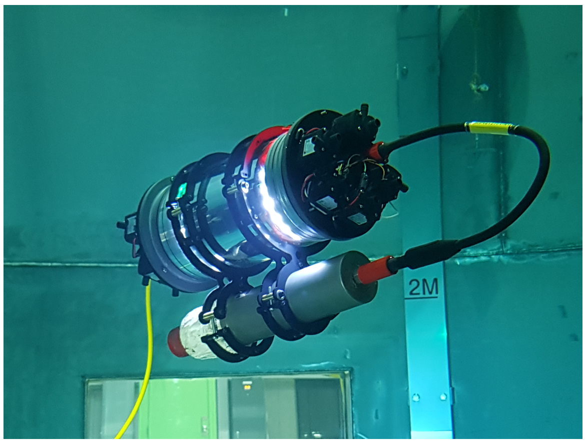
5.2 Potential Future Work 47

Bibliography 49

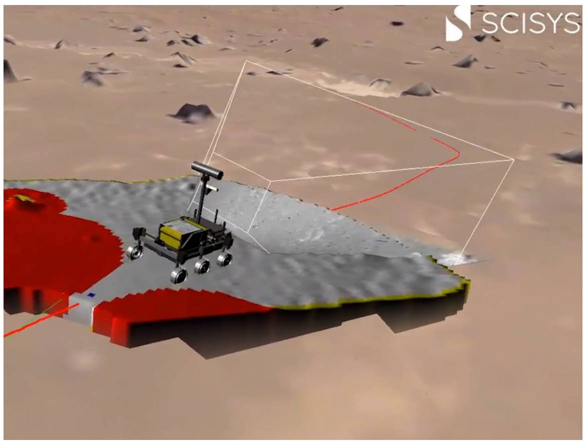
Appendices

Risk Assessment . . . . a

1 Introduction



(a) The AVEXIS submersible robotic vehicle, developed to investigate radioactive debris. Image used with permission of The University of Manchester.



(b) Visualisation of a Mars rover prototype navigating a challenging environment. Image: ESA / SCISYS / UKSA.

Figure 1.1: Two examples of mobile robotics applications where safe planning is important.

1.1 Background and Project Aims

For many tasks that autonomous mobile robots are well suited to, the robotic agent may need to plan and navigate in an environment where there is an *a priori* unknown distribution of a *hazard*. Examples of hazards include steep terrain (for planetary surface exploration rovers), water depth or current (for underwater autonomous vehicles), or radiation exposure (for disaster recovery/nuclear material inspection robots). We assume that as the robot explores it is able to measure how hazardous its current location is. It should then be able to use this information to make *predictions* about the distribution of the hazard, and carry out *planning* using those predictions, to ensure its future navigation is carried out in a safe manner.

There are many good reasons for ensuring robot safety, outside of maximising the usable lifetime of the robot system (which is particularly key for planetary exploration rovers which are expensive to transport to their destination). Broken-down robots may be difficult to safely retrieve [1, 2] and may therefore cause environmental damage or impede access to an area for future robotic missions. The safety of the robot's behaviour may also directly impact the safety of humans that have to work alongside it.

In this project, we consider exploration across models that can be represented as a *Markov decision process* (MDP). These models are commonly used for planning under uncertainty for robots, and can be generalised to add the ability to handle incomplete information. For example, it can be extended to a Partially Observable MDP (POMDP).

This project takes significant inspiration from previous work in the field of safe exploration - in particular, the family of exploration algorithms based on the SafeMDP algorithm originally proposed by Turchetta et al. [3]. In common with these algorithms, we make use of *Gaussian process* models [4] to predict the safety of states that have yet to be visited. These are relatively simple non-parametric models, popular for their high performance on non-linear regression tasks. There is a significant body of literature available covering their application to a wide range of problems, including the task of modelling the spatial distribution of radioactive material.

On the planning side (an area not addressed in depth by previous safe exploration literature), this work makes use of techniques from formal verification of systems. Concepts from *probabilistic model checking* are used to provide probabilistic guarantees on the agent's ability to fulfil goals safely (such as calculating the probability and associated expected cost of safely navigating to a new location to carry out a measurement).

The overall aim of this project is to investigate and develop methods of handling uncertainty and ensuring safety in an autonomous agent's environment. We make several contributions to the field of safe exploration, proposing a new exploration framework that makes use of new and powerful formalisms and algorithms that were developed during the course of the project.

1.2 Literature Review

Classical approaches to robot safety include specifying in advance which types of actions are unsafe, attempting to define recovery behaviours that should allow the agent to make itself safe again if it detects it has encountered an issue, or limiting the agent to ergodic policies that are sure to let the agent recover from any visited state [5]. However, these classical approaches are generally not scalable, assume that the unsafe states are known *a priori*, or overly restrict the robot's capabilities.

Some more recent approaches are less limited in their utility, as they include a model to make predictions about the safety of the environment. A commonly used model is a Gaussian process (GP) regression, which is particularly useful both for its ability to fit a wide range of functions (depending on the choice of kernel function) and because it readily provides uncertainty bounds on its predictions. GP exploration has been investigated in [6], where the value of an unknown objective function, modelled with a GP, is optimised while avoiding the risk of sampling the function where its value is below a safety threshold. This paper therefore considers a stateless “multi-armed bandit” scenario where the agent’s reward depends only on actions taken, without the added structural constraints of an MDP. The example chosen for this paper was outside the field of mobile robotics, and detailed a medical setting where effective therapies must be explored and exploited, but where the agent must avoid suggesting a therapy whose effectiveness is too low as this may harm the patient.

Building upon the ergodic returnability proposed in [7] and the GP model in [6], Turchetta et al. [3] proposed a method of safely exploring MDPs. They formalised the notions of *reachability* from and *returnability* to the safe set of states in the MDP. We extend several of these concepts in this report.

However, no existing literature tackles the problem of safely exploring an MDP with a probabilistic transition function: [3] and [8] consider only MDPs with a deterministic transition function between states, which restricts the class of problems the algorithms can be applied to.

Wachi et al. [8] consider a safe exploration setting in which the agent is also seeking to maximise an unknown objective function, also estimated by a Gaussian Process model, inside the safe explorable area. They also explore environments where the safety process changes over time, using spatio-temporal GPs [9]. Separately, [10] extends the safe exploration approach of [3] to support goal-driven exploration where the agent has a goal state to reach (provided by a higher-level “oracle”) and aims to carry out the minimum exploration required either to reach the goal or determine that it cannot reach it safely. We do not directly tackle these extension problems in this report (they are left as potential future work), but the techniques presented could be extended in much the same way as in the literature above.

As the focus of this project is on safety constrained problems, the formalisms we develop are based on the safe exploration literature detailed above. However, parallel work has been carried out on similar problems involving exploration and planning under uncertainty, but from a purely Bayesian optimisation perspective where an agent wishes to locate the maximum of an unknown function. This body of work includes [11], which makes use of a POMDP model where the observation function is modelled as a GP. This is similar in concept to our formalism proposed in Section 2.6.1: both roll the GP into the MDP model, whereas previous safe exploration literature only uses the GP to produce worst-case MDPs (Section 2.5) which are then separately solved. As POMDPs are more complex than MDPs, and exact solutions to them are generally intractable, Monte Carlo tree search (MCTS) techniques are used to generate solutions in this paper and related ones.

Along the same lines, Flaspohler et al. [12] make heavy use of information theory to determine the optimal approach to a maximum-seek-and-sample task that is similar to the Bayesian optimisation problem in [8]. They develop MCTS-based planning algorithms to carry out the task, using the POMDP/GP formalism discussed in [11]. Although this paper does not cover safety-constrained problems, its “informative path planning” approach to gathering information is closely related to this project’s goal of identifying safe paths to informative states to visit.

Many of the techniques discussed here are also applicable to a control theory setting in continuous time, and existing literature has some examples of this application. In [13], Koller et al. apply some of the underlying concepts of SafeMDP to a model predictive control setting. In another piece of recent literature, Polymenakos et al. [14] tackle GP multi-step-ahead prediction in the context of control theory. They develop theory to provide formal probabilistic bounds on the GP predictions that can be propagated multiple steps for path planning purposes. These can then be used in a safe reinforcement learning framework to safely learn the parameters of a controller. The task of choosing an optimal controller can be thought of as being analagous to our task of choosing an optimal policy for planning paths.

There is also of course significant interest in ensuring safety for machine learning systems. For example, in reinforcement learning (which also often makes use of an MDP model), significant thought is required to prevent the agent from carrying out exploratory actions that are designed to gather useful data but may endanger the robot. One approach is to add the concept of risk to the agent’s optimisation criterion [5], but this still does not assign a hard limit on how much the agent is willing to risk its safety - and also assumes that the undesirable states are known *a priori*.

In summary, we can identify connections between several strands of recent work across multiple research areas. This highlights the relevance and wide potential applications of work on the topic of safe exploration.

1.3 Structure of this Report

The remainder of this report is split into 3 main sections:

- Chapter 2 covers the theoretical constructs developed during the course of this project. After setting out the preliminary theory, a short section covers additional background theory that is referred to in the experiments. In later sections, the description of the exploration approach is split into two parts: the high-level approach is laid out in Section 2.4, and two different approaches to the path planning algorithms are investigated in Sections 2.5 and 2.6.
- Chapter 3 shows how the data structures and algorithms were implemented, and results of experiments in simulation that were run to validate and test the exploration approach.
- Chapter 4 documents the integration of the exploration software implementation into a robotic system, and the development of other tools that enable testing in a more representative real-world environment than the simulated tasks in the previous chapter.

2 Theoretical Approach

2.1 Preliminaries

2.1.1 Markov Decision Processes

A Markov decision process is a formalism frequently used in robotics to represent a series of sequential decisions, which can have probabilistic outcomes that affect the state of the robot and the environment. An in-depth explanation of techniques for solving MDPs (determining the optimal action(s) to take in any MDP state, typically involving the use of dynamic programming methods) is outside the scope of the report, and is covered in depth by [15].

Definition 1 (MDP)**.** A Markov decision process is defined as a tuple $\mathcal{M} = \langle S, \overline{s}, A, T, C, AP \rangle$, where:

- S is a finite set of states, which we assume we can factor into a set of n *state features* such that $S = X_1 \times \ldots \times X_n$. Examples of state features could be x position, y position, water depth, etc,
- $\overline{s} \in S$ is the distribution over initial states the agent may start in (which may have all probability mass assigned to a single known initial state),
- A is a finite set of actions with pre-conditions (on state feature values) that must be satisfied for an action to be *enabled*, and probabilistic outcomes on how the action will evolve the state,
- $T : S \times A \times S \to [0, 1]$ is a probabilistic transition function representing the possible outcomes of an action, where $\sum_{s' \in S} T(s, a, s') \in \{0, 1\}$ for all $s \in S$, $a \in A$,
  - The set of actions that are enabled in state s is defined as $A_s = \{a \in A \mid \text{ exists } s' \in S \text{ such that } T(s, a, s') > 0\}$,
- $C : S \times A \to \mathbb{R}$ is a cost structure defined across the MDP which specifies the "cost" (or reward, expressed as a negative cost) of taking an action in a state,
- AP is a set of atomic propositions. These are logical formulae on the state feature values that must evaluate to true or false.

This definition can straightforwardly be transformed into other similar definitions of an MDP, such as where the cost is associated with a unique action outcome rather than taking the action itself.

2.1.2 Constrained Reachability Optimisation and Linear Temporal Logic

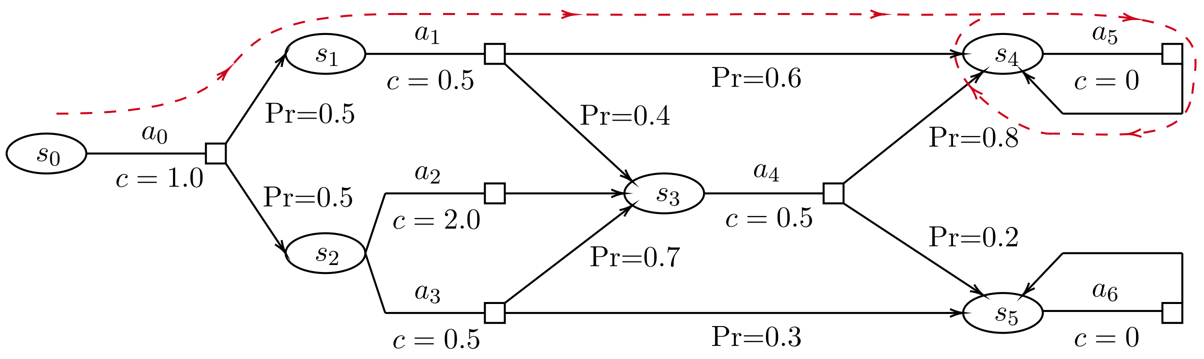


Figure 2.1: A simple MDP showing an example path that has resulted from the actions taken at each state. The path is infinite length as state 4 (as well as state 5) has a self-loop.

Taking actions at each state in an MDP results in the agent taking a *path* through the MDP.

Definition 2 (Path)**.** A path through an MDP $\mathcal{M}$ starting in $s_0 \in S$ is defined as a sequence $w = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} ...$ where $T(s_i, a_i, s_{i+1}) > 0$ for all i. For MDP $\mathcal{M}$ and state $s \in S$, we denote the set of all paths starting in s as $Path_{\mathcal{M},s}$.

The choice of action to take at each step of the execution of an MDP is made by a *policy*.

Definition 3 (Policy)**.** We define a policy over an MDP as a function $\pi : S \to A$ such that, for any state $s \in S$, $\pi(s) \in A_s$. This makes our policies *deterministic* and *memoryless*, as they specify a single action to be taken (rather than a probability distribution over actions) and depend only on the current state s. This form of policy suffices for the problems tackled in this report.

We denote the set of all policies as $\Pi_{\mathcal{M}}$, and the set of paths under a policy π as $Path^{\pi}_{\mathcal{M},s} \subseteq Path_{\mathcal{M},s}$. With this, we can define a probability measure $Pr^{\pi}_{\mathcal{M},s}$ over $Path_{\mathcal{M},s}$, following policy π - this is a probability distribution over the paths that could result from the agent following the policy. Accordingly, we can also define *expected values* $E^{\pi}_{\mathcal{M},s}(\cdot)$ over paths.

Definition 4 (Cumulative Cost Function)**.** We also define a cumulative cost function $cumul$: $Path_{\mathcal{M},s} \to \mathbb{R}$ that maps a path w to the cost accumulated (which is finite only for paths which end in a loop with zero cost):

$$cumul(s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} ...) = \sum_{i=0}^{s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} ...} c(s_i, a_i). \tag{2.1}$$

$E^{\pi}_{\mathcal{M},s}(cumul_{Path_{\mathcal{M},s}})$ therefore refers to the expected cumulative cost across the corresponding probability distribution over paths $Path_{\mathcal{M},s}$.

Definition 5 (Constrained Reachability)**.** In the context of safe exploration, we are particularly interested in identifying a policy to reach one of a set of *goal* states $G \in S$ while avoiding entering any of a set of *forbidden* states $F \in S$, while minimising the cost of doing so. The set of paths that satisfy this desired behaviour are:

$$\begin{aligned} reach_{F,G} = \{(s_0 \overset{a_0}{\to} s_1 \overset{a_1}{\to} \dots \overset{a_{i-1}}{\to} s_i \overset{a_i}{\to} \dots) \in Path_{\mathcal{M},s_0} \mid \text{ exists } i \in \mathbb{N} \\ \text{such that } s_i \in G \text{ and } s_j \notin F \text{ for all } j \leq i\}. \end{aligned} \tag{2.2}$$

The set of policies that are optimal can be expressed as:

$$\{\pi \in \Pi_{\mathcal{M}} \mid \pi = \arg\max_{\pi'} Pr^{\pi'}_{\mathcal{M},\bar{s}}(reach_{F,G})\} \tag{2.3}$$

The overall optimisation objective can therefore be defined as finding the policy $\pi^*_{F,G}$ with the lowest expected cost, out of the policies that maximise the probability of satisfying the constrained reachability problem. Formally:

$$\pi_{F,G} = \arg\min_{\{\pi \in \Pi_{\mathcal{M}} \mid \pi = \arg\max_{\pi'} Pr^{\pi'}_{\mathcal{M},\bar{s}}(reach_{F,G})\}} E^{\pi}_{\mathcal{M},\bar{s}}(cumul_{F,G}). \tag{2.4}$$

Definition 6 (Linear Temporal Logic)**.** Linear temporal logic (LTL) [16] is a specification language which allows logical specifications of goals or tasks as an LTL formula. The LTL formula for the constrained reachability problem above is $(\neg F \text{ U } G)$, where U is the "until" operator.

Co-safe LTL is a subset of LTL that covers task specifications that are able to be completed in a finite time period. The above expression is co-safe. An informal justification for this would be that it does not require an infinite path or loop - the equation is satisfied when the agent reaches one of the states in G, and any further steps are irrelevant.

The constrained reachability problem in Definition 5 is effectively a dual optimisation criterion approach to a stochastic shortest path problem as discussed in [17], where it is also possible that finding a policy that will satisfy the task with probability one is not guaranteed. Lacerda et al. [18] further extend the optimisation problem by encoding it as a co-safe LTL specification and introduce the notion of a task progression metric: the agent will choose a cost-optimal policy from those that maximise expected progression towards completion of the task, when no policy can be found that satisfies the task with probability one. This allows the agent to satisfy as much of the task specification as possible, even when it does not expect to be able to completely satisfy it.

2.1.3 Gaussian Process Regression

The previous section assumes that the forbidden, unsafe set of states is known. However, for our safe exploration task, unvisited states have unknown safety and the agent must have some mechanism to make predictions about the safety of these states. As detailed in the literature review, a Gaussian process (GP) model is a good fit for these requirements, as it also provides measures of predictive uncertainty which can be interpreted as confidence regions on the prediction at a state.

This report denotes scalars in italics (x), vectors in bold type ($\mathbf{x}$), and matrices in capitals (X).

Definition 7 (Gaussian Process)**.** A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution [4]. A GP model is of the form $o(x) \sim \mathcal{GP}(m(x), k(x, x'))$, and represents a probability distribution over functions, fully specified by its mean function $m(x)$ and *kernel* or *covariance* function $k(x, x')$. We can let $m(x) = 0$ without loss of generality.

We wish to carry out a GP regression to predict values $\mathbf{o}(\mathbf{x}_*)$ at a vector of test points $\mathbf{x}_*$, given a vector of noisy data $\mathbf{y}$ where $y_i = o(x_i) + n_i$. For a standard GP regression, the data points are assumed to be related to the underlying function o by a Gaussian noise model likelihood function, with measurement noise $n_i \sim \mathcal{N}(0, \sigma_n^2)$. For a standard GP regression we assume that σ_n is constant across data points, to give a likelihood function of $p(y \mid o) = \mathcal{N}(o, \sigma_n^2)$. It is also possible to define a heteroscedastic GP where the noise variance is different for each data point [4]. The joint distribution of the observed data point values and the function values at the test points can then be written as:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{o}(\mathbf{x}_*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right) \tag{2.5}$$

For the above equation (given n data points and n_* test points), $K(X, X_*)$ denotes the $n \times n_*$ matrix of the covariance function evaluated at all pairs of training and test points, and similarly for the other entries $K(X, X)$, $K(X_*, X)$ and $K(X_*, X_*)$. $I \in \mathbb{R}^{n \times n}$ is the identity matrix.

Conditioning this joint Gaussian prior distribution on the observations y gives the (predictive) posterior distribution, which is another multivariate Gaussian distribution:

$$\mathbf{o}(\mathbf{x}_*) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \text{ where:} \tag{2.6}$$

$$\text{Mean:} \quad \boldsymbol{\mu} = K(X_*, X)\left[K(X,X) + \sigma_n^2 I\right]^{-1} \mathbf{y} \tag{2.7}$$

$$\text{Covariance:} \quad \boldsymbol{\Sigma} = K(X_*, X_*) - K(X_*, X)\left[K(X,X) + \sigma_n^2 I\right]^{-1} K(X, X_*) \tag{2.8}$$

$$\text{Mean (with } m(x) \neq 0\text{):} \quad \boldsymbol{\mu} = \mathbf{m}(X_*) + K(X_*, X)\left[K(X,X) + \sigma_n^2 I\right]^{-1} (\mathbf{y} - \mathbf{m}(X)) \tag{2.9}$$

Equation (2.9) covers the case where the GP has a fixed, deterministic mean function rather than assuming it to be zero. This is straightforward to include as the usual zero mean GP is simply used to model the difference between the data point values and the deterministic mean function.

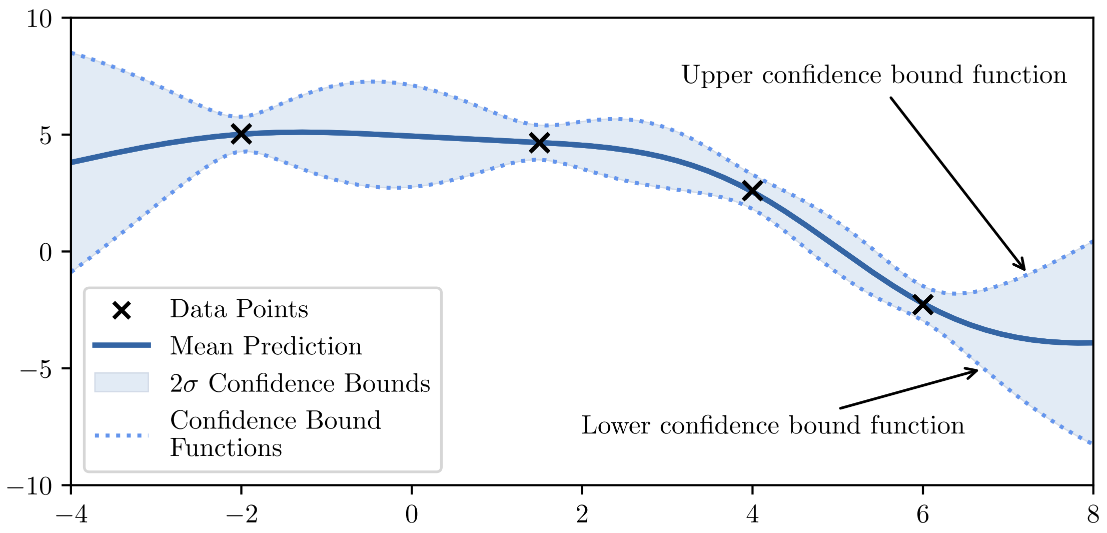


Figure 2.2: An example of a Gaussian process regression with noisy data points, showing $\pm 2\sigma$ confidence bounds around the predicted mean function (equivalent to a ~95% confidence interval).

For clarity in this report, we will also write the input of a GP function as a state s rather than explicitly listing state feature variables. This is equivalent to the input vector of the GP being composed of the relevant state feature values. For example, if the GP is defined over the x position state feature and the y position state feature of a state, the input of the GP model will be a 2-dimensional vector $\begin{bmatrix} x \\ y \end{bmatrix}$.

The result of a GP regression is a posterior probability distribution over admissible functions that could have resulted in the data points, which can be expressed as a mean function and a confidence interval (for a specified confidence level), as shown in Figure 2.2 for a 95% confidence region. Functions can be sampled from the GP posterior distribution by sampling the GP at an arbitrary number of points.

Each point of the function is distributed as a Gaussian, and these confidence intervals are symmetrical around the mean prediction and are specified by the standard deviation of the Gaussian as expected.

We refer to the functions that make up the confidence interval boundaries as the lower and upper *confidence bound functions*. Clearly, these functions depend on the number of standard deviations specified by the chosen confidence interval.

The covariance matrix of the multivariate Gaussian distribution produced by the GP regression is a result of the GP kernel function chosen. The kernel function effectively encodes regularity assumptions about the similarity of unknown states that are close to each other. Many families of kernel functions exist, which can also be combined to form more complicated kernel functions. For example, a kernel function with both a linear and a periodic element might be a good choice of kernel function for a tidal system with long-term changes in tide amplitude. We mainly make use of two kernel functions in this report: the RBF kernel (2.10) and the Matérn 5/2 kernel (2.11).

$$k(x,x') = \sigma^2 \exp\left(-\frac{r^2}{2l^2}\right), \quad (2.10) \qquad k(x,x') = \sigma^2\left(1+\frac{\sqrt{5}r}{l}+\frac{5r^2}{3l^2}\right)\exp\left(-\frac{\sqrt{5}r}{l}\right), \quad (2.11)$$

where $r = \|x - x'\|$.

Kernel functions have hyperparameters such as lengthscale l and variance σ in the kernels above. The values of these hyperparameters may be fixed or assigned a prior distribution to encode prior knowledge about the dynamics of the function being estimated. Any parameters with prior distributions assigned can then be optimised in a maximum likelihood estimation manner, by maximising the log marginal likelihood $\log p(\mathbf{y} \mid X)$ of the data points. This value is obtained by marginalising over the values of the data points:

$$p(\mathbf{y} \mid X) = \int p(\mathbf{y} \mid \mathbf{o}, X) p(\mathbf{o} \mid X) d\mathbf{o} \tag{2.12}$$

This can intuitively be understood as an automatic method to avoid overfitting the model, along the lines of Occam's razor. The maximisation is typically carried out with gradient descent optimisation methods. The amplitude of the Gaussian measurement noise in the likelihood is often considered as another parameter which can be fixed or determined through optimisation. The regression in Figure 2.2 has optimised the length and variance hyperparameters of its RBF kernel (2.10) as well as the measurement noise amplitude. The non-zero measurement noise value determined is the reason for the confidence bounds not collapsing to zero at the data points.

As discussed in existing literature [3, 8], some standard modelling assumptions must be made about the GP model to justify its use in estimating the safety of states in our model. We must assume that o has bounded norm in the Reproducing Kernel Hilbert Space (RKHS) associated with the chosen kernel function, and also that it is Lipschitz continuous with respect to some metric $d(\cdot,\cdot)$ on S. Having bounded norm in the RKHS ensures that the function o is smooth with respect to the kernel, so that the GP can justifiably be expected to be able to model it with the specified kernel.

The Lipschitz continuity requirement effectively means that the GP's maximum rate of change (which can readily be thought of as a spatial gradient for a GP with a 1- or 2-dimensional input) is greater than the maximum rate of change expected of the underlying function o being modelled:

$$\mid o(\mathbf{x}_1) - o(\mathbf{x}_2) \mid \; \leq \; L \cdot d(\mathbf{x}_1, \mathbf{x}_2) \tag{2.13}$$

For the kernels used in this report this relates to the lengthscale chosen as well as the choice of kernel function - if a too-long lengthscale is chosen for modelling, the GP prediction uncertainty will increase slowly with distance from measured data points. This will not capture the real behaviour of the safety feature function, which may have diverged significantly from the predicted confidence bounds. On the

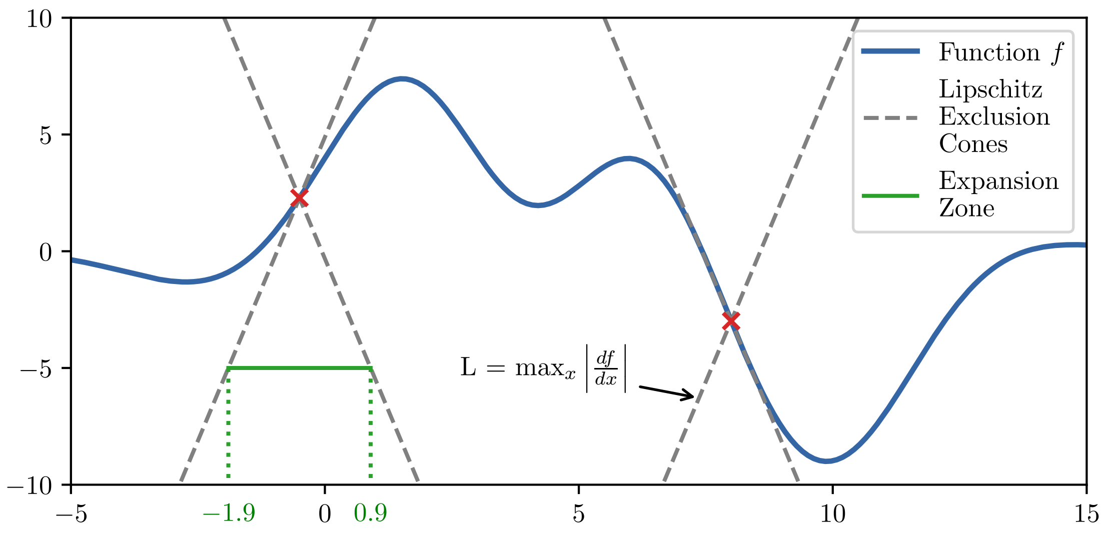


Figure 2.3: A simple 1-dimensional graphical interpretation of Lipschitz continuity. The Lipschitz constant L (which is ≈ 5.2 for this plot) effectively defines exclusion cones where function values cannot lie. The expansion zone concept, that effectively makes use of the Lipschitz exclusion cones, is discussed in Section 3.2.4.

other hand, if a much shorter lengthscale is chosen, the certainty bounds on the GP's predictions will diverge more rapidly with distance and the GP's predictions will be less useful for exploration.

This is shown graphically for one dimension in Figure 2.3. Turchetta et al. [3] make use of the Lipschitz constant for theoretical proofs on the performance of their exploration algorithm, as well as an additional step in the SafeMDP algorithm as described in Section 3.2.4.

The overall requirement that the kernel function chosen is able to accurately model the dynamics of *a priori* unknown underlying hazard features in the environment may appear to be overly restrictive. However, the necessity of assuming some regularity of the underlying function is inherent to this type of exploration problem - it is required to make the exploration problem tractable.

Similarly, the chosen rate of exploration (i.e. the distance one is willing to travel between taking measurements) is always tied to the maximum expected rate of change (i.e. the Lipschitz constant) of any environmental feature that affects safety. If one was trying to navigate a cloud of harmful gas, one would check a gas meter much more frequently if the concentration of gas was expected to change on the order of centimeters rather than on the order of metres.

For most applications, there will be at least some indication of the expected spatial gradient of hazards - as an example, the steepest possible gradient in radiation level for a radioactive environment will be the inverse square $\propto \frac{1}{(\text{distance to source})^2}$ [19] distribution for a single, concentrated source. Although this gradient of this function tends to infinity as (distance to source) $\to 0$, there will be a minimum distance between the robot and source - either defined by physical geometry or how close the robot is safely willing to go.

2.2 Additional Background Theory

2.2.1 Kullback–Leibler Divergence Between Two Multivariate Normal Distributions

The Kullback–Leibler (KL) divergence is a useful metric to measure the similarity between two probability distributions, and is defined as in (2.14) for continuous probability distributions P and Q.

$$D_{KL}(P_1 \parallel P_2) = \int_{-\infty}^{\infty} p_1(x) \log\left(\frac{p_1(x)}{p_2(x)}\right) \tag{2.14}$$

We are particularly interested in calculating the divergence between the posterior (predictive) distributions of two GPs, sampled at the same n points. These are multivariate normal distributions (MVNs) in $\mathbb{R}^n$, with the mean vector being the predicted mean of each sample point, and the covariance matrix consisting of the predicted variance of each sample point, along the diagonal of the matrix. The KL divergence for two MVNs $P_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$ and $P_2 \sim \mathcal{N}(\boldsymbol{\mu}_2, \Sigma_2)$, derived in Section 9 of [20], is given in (2.16). It is clear from (2.15) that if $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2$ and $\Sigma_1 = \Sigma_2$, then the calculated KL divergence is zero as the distributions are the same.

$$D_{KL}(P_1 \parallel P_2) = \frac{1}{2}\left(\log\frac{\det\Sigma_2}{\det\Sigma_1} - n + \operatorname{tr}\left(\Sigma_2^{-1}\Sigma_1\right) + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^{\mathsf{T}}\,\Sigma_2^{-1}\,(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)\right) \tag{2.15}$$

To make this calculation manageable for large n (for comparing distributions across many sample points), we can make use of the fact that Σ_1 and Σ_2 are diagonal matrices:

$$D_{KL}(P_1 \parallel P_2) = \frac{1}{2}\left(\sum_{i=1}^{n}\log\frac{\Sigma_{2,ii}}{\Sigma_{1,ii}} - n + \sum_{i=1}^{n}\frac{\Sigma_{1,ii}}{\Sigma_{2,ii}} + \sum_{i=1}^{n}\frac{(\mu_{2,i} - \mu_{1,i})^2}{\Sigma_{2,ii}}\right) \tag{2.16}$$

2.2.2 Poisson Statistics of Radiation Exposure

Several experiments in this report concern the modelling of radioactive environments. Radioactive decay is well known to be a Poisson process [21], and in the same way the occurrence of radioactive *events* in time causing corresponding *counts* on a measurement instrument can be modelled by the Poisson distribution. Equation 2.17 defines the probability mass function for an instrument detecting k events in a time period given the expected number of occurences per time period λ (which is defined by the actual radiation level multiplied by any dimensionless geometric factors).

$$f(k, \lambda) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{2.17}$$

The central limit theorem can be used to give the normal approximation to the Poisson distribution for large λ:

$$\operatorname{Pois}(\lambda) \approx \mathcal{N}(\lambda, \lambda) \tag{2.18}$$

2.3 Exploration Model

Previous approaches to safe exploration in MDPs [3, 7, 8, 9, 10] separate the safety dynamics (usually represented as a single scalar-valued function) from the standard MDP model. The initial goal of the project was to re-formalise the safe exploration problem so that the safety of states is more closely connected to the state representation of the MDP model. This would provide a richer underlying model that would make the modelling approach feasible for more complex problems than those previously tackled in literature, support the development of better-performing exploration algorithms, and allow for richer specifications over the unknown safety features. The exploration model formalisms developed during the course of the project are presented in this section.

Definition 8 (MDP with Unknown Feature Values)**.** An MDP with Unknown Features Values (U-MDP) $\mathcal{M}^o = \langle S^o, \bar{s}, A, T^o, c \rangle$ is defined as an MDP where the state space is factorised as $S^o = S_k \times S_e$, where S_k is the set of *known* state features ($S_k = X_{k,1} \times X_{k,2} \times ... \times X_{k,n}$) and S_e is the set of *unknown* state features ($S_e = X_{e,1} \times X_{e,2} \times ... \times X_{e,n}$).

Furthermore, there is an unknown mapping $o : S_k \to S_e$ that defines which unknown state feature $o(s_k) \in S_e$ is observed at $s_k \in S_k$. In other words, the state of the U-MDP is defined as $(s_k, o(s_k)) \in S_k \times S_e$, but o is unknown *a priori*.

The U-MDP (and the mapping function o) is a discrete representation of an underlying continuous state space - o is a discrete mapping function, but its underlying continuous domain function can be estimated by noisy sensor readings. Section 2.3.3 discusses how we translate between the continuous and discrete domains.

Finally, given that a state is uniquely defined by the value of the known state feature $s_k \in S_k$, the outcome of the transition function only represents the change in the known state feature. Formally, $T^o : (S_k \times S_e) \times A \times S_k \to [0, 1]$, where $T^o((s_k, s_e), a, s'_k)$ represents the probability of moving to state $(s'_k, o(s'_k))$ given that action a was taken at state $(s_k, o(s_k))$.

We also define a safety function over the state features as $g : S \to \{0, 1\}$ where $g(s) = 1$ when s is considered safe. An example of a simple safety function is an upper-bound threshold $b \in \mathbb{R}$ on an unknown state feature. The sets of safe and unsafe states can then be defined:

$$\mathit{safe} = \{(s_k, s_e) \in S_k \times S_e \mid g(s_e) = 1\} \tag{2.19}$$

$$\mathit{unsafe} = (S_k \times S_e) \setminus \mathit{safe} \tag{2.20}$$

This formalisation offers richer modelling than previous approaches in the following ways:

- The formalisation provides for MDPs with probabilistic transition models;

- Posing more of the exploration problem in the form of an MDP also enables parts of it to be tackled with existing tools such as probabilistic solvers that work with LTL specifications;
- More complex safety specifications are supported than the simple upper-bound on a scalar value used in previous literature;
- The dynamics of the known state features may depend on the value of unknown state features. This allows for applications where the same physical process affects both safety and transition behaviour: for example an underwater autonomous vehicle's safety may depend on the strength of water currents, and these currents also directly affect its motion when it tries to navigate; and
- It opens up potential connections between work in the field of safe exploration and work from a Bayesian optimisation perspective such as [12]. The U-MDP can be reformulated as a POMDP, but for our goal of safe exploration this is not necessary.

We refer to combinations of known state features S_k as *known states* s_k, and the safety of these states is determined by the corresponding unknown state feature values S_e that the known state maps to, defined by $o(s_k)$. Exploration is therefore across known states - it is not possible to visit all states $(S_k \times S_e)$ of the U-MDP due to the one-to-one mapping between known states features and unknown state features.

2.3.1 Reachability and Returnability in an MDP

As well as the basic definition of which states are safe and which are unsafe, the MDP structure introduces additional safety-related properties that must be considered to generate safe policies across the MDP. As introduced by [3], the agent must also consider two concepts that are defined by the MDP transition structure: the notions of *reachability* and *returnability*.

A state is reachable if it is possible for the agent to transition to the state (possibly in multiple steps) without entering an unsafe state, starting in the set of states that it has already explored - the *current explored set*. Similarly, a state is returnable if it is possible to return from the state to the current explored set (possibly in multiple steps) without entering an unsafe state.

These two conditions ensure that the exploring agent does not choose to visit a state that there is not a safe route to, and does not become trapped in an 'island' of safe sets without any way to return and continue exploring. If the agent only explores states which satisfy these two requirements then the currently explored set will always consist of states that are reachable and returnable from each other. Additionally, this current explored set must consist of states that are safe - as the agent has already visited each of them (and had they not been safe, the agent would already have failed its task).

2.3.2 Formal Problem Definition

Other than ensuring the safety of the robot, the desirable outcomes for the completed exploration are to gain knowledge of the dynamics of the safety feature and to determine to a high degree of confidence the set of states that are safe, reachable and returnable.

The safe exploration problem can now be defined as:

Problem definition: Safe Exploration in a U-MDP

Given a safety function $g()$ over the U-MDP state features, minimise the expected cumulative cost to determine the mapping $o(s_k)$ to a given accuracy ϵ for all states in the safely reachable and returnable set of states (from the agent's initial state), without visiting a state in *unsafe*.

The requirement to determine $o(s_k)$ to ϵ accuracy only across the predicted safe set is necessary as the agent is not able to sample unsafe states, so the predictive uncertainty of areas of unsafe states may be impossible to reduce.

2.3.3 Mapping from Continuous GP Domain to Discrete MDP States

In subsequent sections, predictions from GP models will be used to estimate the mapping $o : S_k \to S_e$. However, the GP prediction is a continuous function over $\mathbb{R}$ whereas the unknown state component of the U-MDP consists of a finite number of discrete unknown state features.

We therefore specify that a value of an unknown state feature X_e in S_e is a finite partition of $\mathbb{R}$, given as $X_e = \{I_1, \ldots, I_p\}$ where I_i are non-overlapping intervals of $\mathbb{R}$ such that $\cup_{i=1,\ldots,p} I_i = \mathbb{R}$.

For example, a GP model estimating water depth could map to an unknown state feature for water depth in metres, where the intervals are $\{(depth < 10), (10 \leq depth < 20), (20 \leq depth)\}$. This would result in the water depth state feature having 3 valid discrete values. Each unknown state feature value is equivalent to an interval I. An estimation of $o(s_k)$ would therefore map each known state (which could be a combination of x position and y position known state features) to the water depth interval at that location.

A potential area for future work is a more in-depth investigation into the discretisation, and how this can be carried out while still maintaining important properties of the underlying continuous distributions. This topic is discussed in [22] and [23].

The choice of interval definitions requires some thought, depending on the exploration task. If the safety specification function is a simple upper or lower bound with value b on one or more unknown state features, and the exploration approach from Section 2.5 or Section 2.6 is used, then the most efficient approach is to partition each state feature into two intervals as $\{[-\infty, b), [b, \infty]\}$. Reducing the state space dimensionality makes the constrained reachability problem (2.4) easier to solve.

The choice of intervals does not affect the resolution of the data collected during the exploration, as the GP regression itself still provides the continuous estimation of the state features in the environment. However, reducing the number of intervals gives a less finely grained discretisation and therefore reduces the resolution of the data available to the model solver that solves the constrained reachability problem. This reduces the effectiveness of techniques that carry out more complex path calculations, such as the approach proposed in Section 2.7.

2.4 Top-Level Exploration Algorithm for Probabilistic Transition Models

The first task identified for the project was to build upon previous safe exploration algorithms covered in the literature review section - in particular, the SafeMDP exploration algorithm proposed by Turchetta et al. [3]. These should be extended to be applicable to MDPs with non-deterministic transition functions, while making use of the new formalism (Definition 8: U-MDP).

For clarity of this report, we split the overall exploration algorithm into two parts. This section details the high-level exploration loop where the agent carries out its exploration task using a provided timestep-dependent policy and goal. Sections 2.5.1 and 2.6.2 present different variants of the algorithm used for generating goal states and policies.

2.4.1 U-MDP Exploration Algorithm

The exploration flow diagram (Figure 2.4) provides a high-level overview of the exploration approach, which is given in more detail by Algorithm 2.1. To start, a U-MDP to be explored is provided along with observations of the starting explored set and initial estimates of the covariance structure in the form of kernel hyperparameters. The starting set of observed states must be safe, and each state must be reachable and returnable (Section 2.3.1) to the other states in the starting set. Without this starting set there is not sufficient information to begin exploration.

As well as requiring a starting set, the performance of the exploration algorithm depends on several interactions between the GP kernel, the layout of the discrete states in space, and the dynamics of the unknown state features. These are discussed in depth in Section 2.1.3, but the key requirement is that the GP kernel chosen (including the hyperparameter priors set on the kernel) is able to fully capture the dynamics of the unknown state features.

Similarly to other safe exploration algorithms, the core of the exploration approach is for the agent to use its current knowledge of the U-MDP to determine which *goal* state it should next visit in order

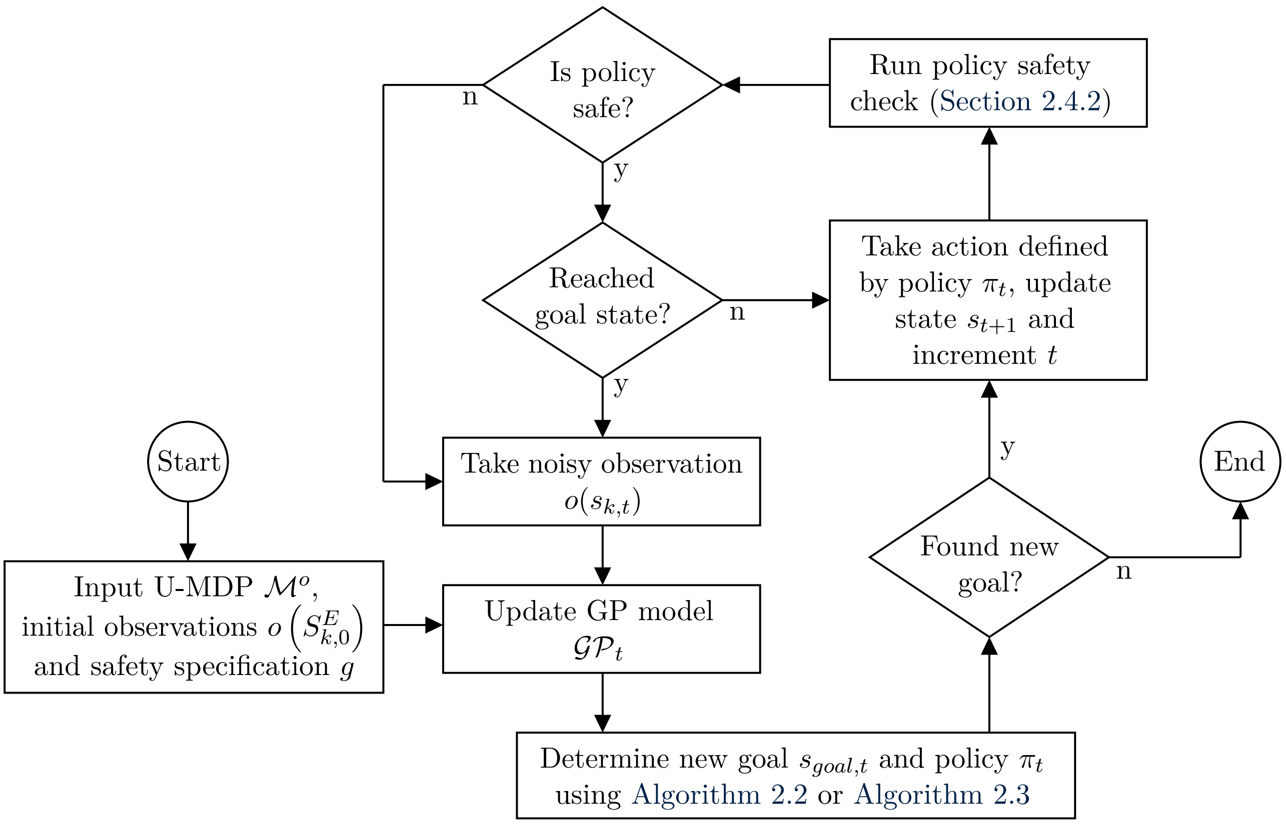

Start
Input U-MDP $\mathcal{M}^o$, initial observations $o\left(S^E_{k,0}\right)$ and safety specification g
Update GP model $\mathcal{GP}_t$
Determine new goal $s_{goal,t}$ and policy π_t using Algorithm 2.2 or Algorithm 2.3
Found new goal?
n
End
y
Take action defined by policy π_t, update state s_{t+1} and increment t
Run policy safety check (Section 2.4.2)
Is policy safe?
n
y
Reached goal state?
n
y
Take noisy observation $o(s_{k,t})$

Figure 2.4: Flow diagram of high-level U-MDP exploration algorithm.

Algorithm 2.1 Safe Exploration of a U-MDP

Input: U-MDP $\mathcal{M}^o$, safety function g, kernel $k(s, s')$, start observations $o(S^E_{k,0})$
Output: Explored U-MDP - in the form of an estimation of $o(s)$

```
1: $s_t \leftarrow \bar{s}$, $s_{goal,t} \leftarrow s_{k,t}$
2: for t = 1,2,.. do
3:     if $s_{k,t} = s_{goal,t}$ or PolicyCheck$(s_{k,t}, \pi_t) > (1 - P_{\min})$ then
4:         $S^E_{k,t} \leftarrow S^E_{k,t-1} \cup \{s_{k,t}\}$, $S^U_{k,t} \leftarrow S_k \setminus S^E_{k,t}$
5:         Observe $o(s_{k,t})$ (with noise), update and optimise $\mathcal{GP}_t$
6:         Get a new goal state
7:     end if
8:     Take action $\pi_t(s_t)$ and update $s_{t+1}$ according to observed outcome
9: end for
```

to best improve its knowledge. When the agent reaches its current goal, or a policy safety check (Section 2.4.2) fails, then an observation is carried out at the current state and a new goal and policy are chosen. The policy safety check is carried out after each action to check whether the outcome of the action has unexpectedly made the agent sufficiently unsafe that it should re-plan its current goal.

s_t is the agent's state at timestep t, and $s_{k,t}$ is the known state component of that state. π_t and $s_{goal,t}$ are the agent's policy and goal at timestep t, respectively. $S^E_{k,t}$ is the set of known states that have been visited by the agent (the *explored set*), and $S^U_{k,t}$ is the remainder of known sets (the *unexplored set*). These sets are updated in line 4, by adding the newly observed state to the explored set and removing it from the unexplored set. When a new goal is chosen, it is chosen from $S^U_{k,t}$. Note that although t is used as an index, the terms above do not necessarily change on each increment of t, as t is the timestep increment determined by how many actions the agent has carried out and these sets, policies and goal states do not necessarily change after every agent action.

When new observations are carried out (line 5), the GP model is updated with the observation and its hyperparameters are optimised. The new GP model $\mathcal{GP}_t$ (with all observations up to this timestep t) is made available to the goal and policy generation algorithm.

Computing the GP regression and generating new goals and policies is computationally intensive. The algorithm therefore aims to observe and re-plan only when it is necessary for exploration. If the time taken to carry out an observation is negligible, then it would be possible to do this while travelling between states rather than only observing the function at goal states. However, in many scenarios there is a time requirement on collecting each observation.

We also define a minimum safety probability measure P_{min}. This is designed to specify the level of safety certainty adhered to while exploring. We typically set it to $P_{min} = 0.95$ to correspond to the common choice of 2 standard deviations of certainty used in other safe exploration literature. Increasing P_{min} increases the certainty that states classified as safe are truly safe to visit, but also may lead to the agent being too pessimistic and avoiding states that are in fact safe.

2.4.2 Policy Safety Check

The *policy safety check* PolicyCheck$(s_{k,t}, \pi_t)$ estimates the probability of the agent ending up in an unsafe state, i.e. a state that does not satisfy the *safe* constraint, within the next n timesteps, while executing its current policy π_t starting in the current state s_t. If the calculated probability is greater than $(1 - P_{\text{min}})$, the agent abandons its current goal, takes a measurement and continues. This check is designed to handle situations where the probabilistic transition model of the MDP may cause the agent to "accidentally" fall into a state where it must generate a new policy to safely leave that state.

Given an MDP, this is a standard property that can be calculated using (2.21). The MDP structure used for this calculation differs for each of the approaches in Sections 2.5 and 2.6 and is described in detail in those sections.

$$\text{PolicyCheck}(s, \pi, n) = \begin{cases} 1, & \text{if } g(s) = 0, \text{else} \\ 0, & \text{if } n = 0, \text{else} \\ \sum_{s' \in S} T(s, \pi(s), s') \cdot \text{PolicyCheck}(s', \pi, n-1), & \text{otherwise} \end{cases} \tag{2.21}$$

The choice of how many steps n to check the policy safety over should be chosen based on a number of factors. If the MDP has many actions at each state then the number of states that have to be checked increases exponentially with the number of steps. On the other hand, if the MDP transition structure means that it is possible for the agent to get “stuck” on a path of states leading to an unsafe area, then a larger n will allow it to see this danger coming more easily.

2.4.3 Probabilistic Determination of Reachability and Returnability

As well as the requirement that the state be safe, SafeMDP [3] limits its search for a new goal state to states that fulfil the reachability and returnability criteria defined in Section 2.3.1. With deterministic transition functions, SafeMDP can simply determine whether states satisfy these criteria using Dijkstra’s algorithm. It also restricts its reachability check to only consider states that can be reached in a single step from the explored set - although the return path can take an arbitrary number of steps.

In the next two sections, we develop algorithms for determining a new goal state, in which we will also wish to carry out reachability and returnability checks. To support probabilistic transition models (which SafeMDP does not), we must define reachability and returnability to/from a state as probabilities that take into account the probability of falling into an unsafe state at each step taken towards/back from the state. In the goal choice algorithms, these probabilities are compared to P_{min} to determine whether the agent considers the reachability and returnability probabilities of a state to be sufficiently high. Reachability and returnability probability calculations are based on constrained reachability optimisation problems and are defined for each approach in Sections 2.5 and 2.6.

2.5 Confidence Bounds Approach to Path Planning

Several safe exploration approaches [3, 8] make use of the GP confidence bounds on the safety function to determine which states are safe, to a certain degree of confidence. For a task with a safety specification that the safety function value must be above a certain threshold, the lower confidence bound function of the GP prediction would be used. All states where the lower confidence bound prediction is below the safety threshold would be classified as unsafe. An identical approach using the upper confidence bound would be used if the agent was instead required to maintain the safety function value below a specified upper limit.

Figure 2.5 shows an example of the bounds approach to safety classification for a lower-bound safety specification. In this example, the agent should not choose to attempt to visit any of the states in the right-most green safe area - although they are considered safe, they are not reachable or returnable to/from its current explored set, without passing through an area of states considered to be unsafe.

The confidence bound functions depend on the the choice of bound certainty value - often taken to be 2 standard deviations for 95% certainty on classifying states as safe. This is equivalent to the definition of P_{min} for our algorithms. Section 3.2.1 discusses the implementation and results of the methods in this section.

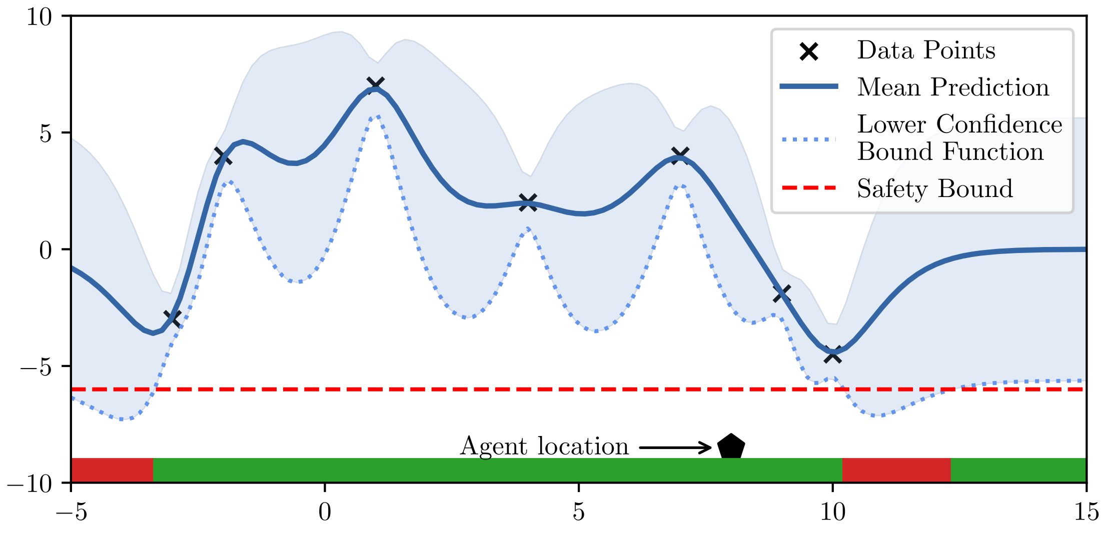


Figure 2.5: Illustration of the confidence bound concept of safe set determination. The colour of the bottom bar indicates whether the agent considers states at that x location safe (green) or unsafe (red).

In this section, we model the exploration U-MDP as a confidence bound MDP (CB-MDP). We do this by estimating $o(s_k)$ (which is a deterministic mapping in the U-MDP) with a deterministic mapping based on our "worst-case" assumptions about the which value of S_e will be observed at s_k, according to the current GP model $\mathcal{GP}_t$. This results in a probabilistic transition function for the CB-MDP that augments the known probabilistic transition function of the U-MDP $\mathcal{M}^o$ to add the transition to the corresponding unknown state for the known state being transitioned to.

This probabilistic transition function encodes the dynamics of both the known and unknown state features. For a given timestep t, we define $\mathcal{CB}_t : S_k \to S_e$ such that $\mathcal{CB}_t(s_k) = I$ where I is the interval containing the value of the appropriate confidence bound function (upper or lower as described above) predicted at s_k.

Definition 9 (Confidence Bound MDP)**.** Given U-MDP $\mathcal{M}^o$ and $\mathcal{CB}_t : S_k \to S_e$ at timestep t, the CB-MDP at timestep t is defined as $\mathcal{M}_t^b = \langle S_k \times S_e, \overline{s}, A, T_t^b, C\rangle$ where:

$$T_t^b((s_k, I), a, (s_k', I')) = \begin{cases} T^o((s_k), a, s_k'), & \text{if } \mathcal{CB}_t(s_k') = I', \\ 0 & \text{otherwise} \end{cases} \tag{2.22}$$

2.5.1 Confidence Bound Goal and Policy Generation Algorithm

Algorithm 2.2 is similar to the goal selection component of SafeMDP [3], but has been augmented to make use of the new U-MDP formalism and to use a CB-MDP to determine which goal state the agent should choose next in order to explore. As well as its use for carrying out reachability and returnability checks, the current CB-MDP model (the model obtained from the GP with the current data points) is used for the policy safety check (Section 2.4.2).

Algorithm 2.2 CB-MDP Goal State Choice

```
Input: CB-MDP $\mathcal{M}_t^c$, $S_{k,t}^E$, $\mathcal{GP}_t$, current state $s_t$, $g$, $k(s,s')$, unexplored set $S_{k,t}^U$
Output: New goal state $s_{goal,t}$, new policy $\pi_t$
 1: $S_{cand} \leftarrow \{s_k \in S_{k,t}^U \mid g((s_k, \mathcal{CB}_t(s_k))) = 1$ and $\Sigma_t(s_k) > \epsilon\}$
 2: while $S_{cand} \neq \emptyset$ do
 3:     $s_{cand} \leftarrow \arg\max_{s \in S_{cand}} \Sigma_t(s)$
 4:     $p_{reach} \leftarrow Pr^{\max}_{\mathcal{M}_t^b, s_t}(reach_{unsafe,\{(s_{cand},I) \mid I \in S_e\}})$
 5:     $p_{return} \leftarrow Pr^{\max}_{\mathcal{M}_t^b, s_{cand}}(reach_{unsafe,\{(s_k,\mathcal{CB}_t(s_k)) \mid s_k \in S_{k,t}^E\}})$
 6:     if $p_{reach} > P_{\min}$ and $p_{return} > P_{\min}$ then
 7:         $s_{goal,t} \leftarrow s_{cand}$
 8:         return $s_{goal,t}$, and corresponding policy $\pi_{unsafe,\{(s_{goal,t},I) \mid I \in S_e\}}$
 9:     end if
10:     Remove $s_{cand}$ from $S_{cand}$
11: end while
12: No goal state identified, end exploration.
```

Intuitively, a “good” goal state should provide information when observed (i.e. have a high GP predictive variance before observation). The goal state chosen is the state with highest GP predictive variance that fulfils three criteria: safety of the state itself, reachability safety and returnability safety.

In line 1, the algorithm restricts the set of candidate goal states to states that are considered safe given the worst-case confidence bound prediction, and that we do not already have good knowledge of. If a state s has GP predictive variance $\Sigma_t(s) < \epsilon$ (where the GP is $\mathcal{GP}_t$, given the current data points at time t) then it is not useful to visit according to the safe exploration problem formulation.

Reachability and returnability probability calculations are carried out on candidate states with the highest GP predictive variance - these calculations are based on constrained reachability problems over the current CB-MDP with the forbidden set defined as the set of unsafe states according to the safety function, taking the calculated satisfaction probability as the corresponding probability. We assume here that the unit of the cost structure of the U-MDP is the time required to carry out the action, but this depends on what exactly the robot system is trying to minimise when it carries out path planning.

For the reachability check (line 4), the constrained reachability optimisation problem is set up with the initial state as the current state s_t, and the goal state defined as the in the CB-MDP where the known component is the candidate state s_{cand}. As well as storing the satisfaction probability, the policy from the solution of the reachability problem is also stored as this is the correct policy to be returned for the agent to use to reach the new state - if this candidate state is chosen as the goal state. How these terms relate to the solution of the constrained reachability problem is explained in Table 2.1.

For the returnability check (line 5), the constrained reachability optimisation problem is set up with the initial state as the candidate state s_{cand}, and the goal set defined as the already explored (hence surely safe) states. If the two probabilities are greater than P_{min} when compared on line 6, then the candidate state is returned as the new goal state along with the corresponding policy to reach it. If no new goal state can be identified, then the exploration terminates in line 12 as specified in the problem definition in Section 2.3.2.

2.6 Estimated MDP Approach to Path Planning

In this section, we model the exploration U-MDP as a *Estimated MDP* (Est-MDP). This is a new formalism that was proposed and developed extensively during the course of this project. It is designed to be a richer underlying model than the confidence bound function MDPs used in previous literature, by directly incorporating the agent's current GP estimates into the MDP model used for planning. We also pair this formalism with a more advanced goal state choice algorithm that scores potential goal states on several criteria rather than solely on GP predictive uncertainty. Section 3.2.2 discusses the implementation and results of the methods in this section.

2.6.1 Estimated MDP Structure

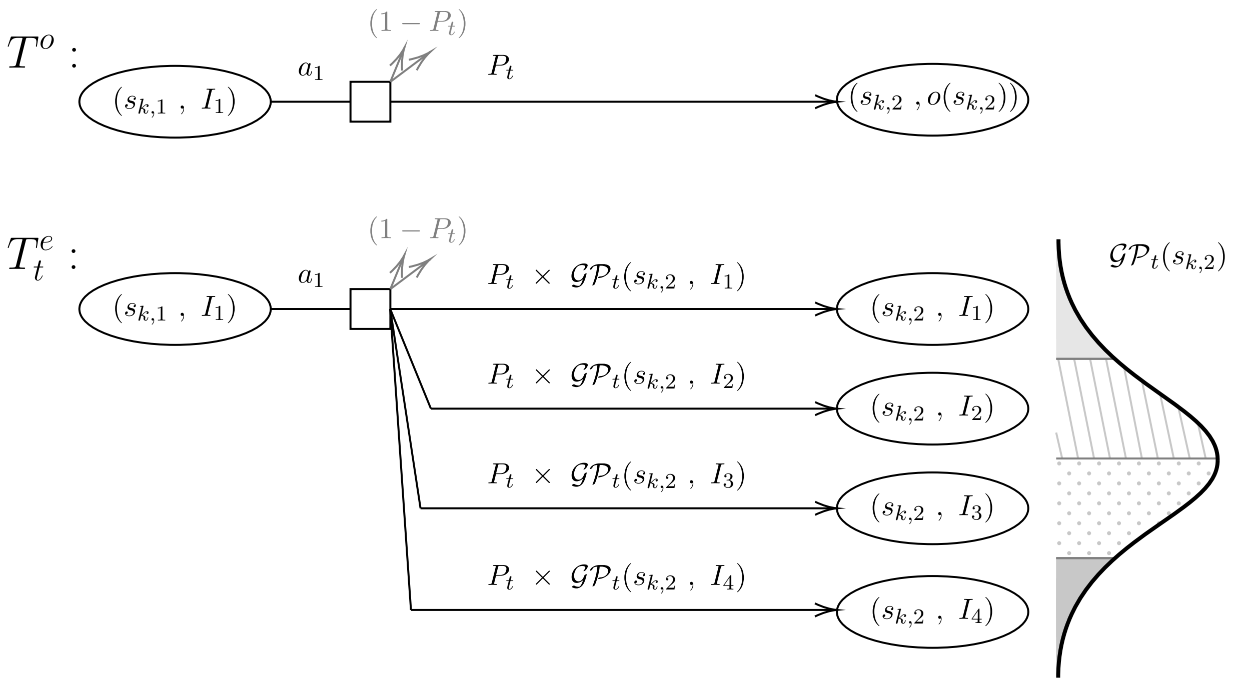

T^o :
$(1-P_t)$
a_1
P_t
$(s_{k,1}\ ,\ I_1)$
$(s_{k,2}\ , o(s_{k,2}))$
T^e_t :
$(1-P_t)$
a_1
$(s_{k,1}\ ,\ I_1)$
$P_t\ \times\ \mathcal{GP}_t(s_{k,2}\ ,\ I_1)$
$P_t\ \times\ \mathcal{GP}_t(s_{k,2}\ ,\ I_2)$
$P_t\ \times\ \mathcal{GP}_t(s_{k,2}\ ,\ I_3)$
$P_t\ \times\ \mathcal{GP}_t(s_{k,2}\ ,\ I_4)$
$(s_{k,2}\ ,\ I_1)$
$(s_{k,2}\ ,\ I_2)$
$(s_{k,2}\ ,\ I_3)$
$(s_{k,2}\ ,\ I_4)$
$\mathcal{GP}_t(s_{k,2})$

Figure 2.6: Example combination of the probabilistic transition function and the current GP distribution over a single unknown state feature variable to create the Est-MDP transition structure.

We model the U-MDP by estimating $o(s_k)$ (which is a deterministic mapping in the U-MDP) with a probability distribution $\mathcal{GP}_t(s_k)$ representing our current estimate of the probability of observing the different values of S_e at s_k, according to the current GP model $\mathcal{GP}_t$.

This results in a probabilistic transition function for the Est-MDP that is a result of weighting the known probabilistic transition function of the U-MDP $\mathcal{M}^o$ by $\mathcal{GP}_t(s_k)$. This probabilistic transition function encodes the dynamics of both the known and unknown state features.

For a given timestep t, we define $\mathcal{GP}_t : S_k \times S_e \to [0,1]$ such that $\mathcal{GP}_t(s_k, I)$ is the probability, predicted by the GP taking into account the observations made up to timestep t, of $o(s_k) \in I$. This can also be expressed as $\mathcal{GP}_t(s_k, .)$ which denotes the probability distribution over S_e represented by $\mathcal{GP}_t$.

Definition 10 (Estimated MDP)**.** Given U-MDP $\mathcal{M}^o$ and $\mathcal{GP}_t : S_k \times S_e \to [0,1]$ at timestep t, the Est-MDP at timestep t is defined as $\mathcal{M}_t^e = \langle S_k \times S_e, \overline{s}, A, T_t^e, C \rangle$ where:

$$T_t^e((s_k, I), a, (s'_k, I')) = T^o((s_k, I), a, s'_k)\mathcal{GP}_t(s'_k, I') \tag{2.23}$$

Figure 2.6 illustrates the construction of T_t^e, where integration over the probability density is used to determine the transition probabilities to assign to each interval. The GP model at the current timestep $\mathcal{GP}_t$ is predicting that I_2 and I_3 are the most likely values of the mapping $o(s_{k,2})$.

2.6.2 Estimated MDP Goal and Policy Generation Algorithm

Algorithm 2.3 makes use of the Est-MDP to determine which goal state the agent should choose to explore. As well as its use for carrying out reachability and returnability checks, the current Est-MDP model is used for the policy safety check (Section 2.4.2). As before, a "good" goal state should provide information when observed (i.e. have a high GP predictive variance before observation). However, the agent should also consider the time taken to reach the goal state, and how how certain it is that it is safe to reach it and return from it. None of these factors are considered by the SafeMDP [3] family of algorithms.

Algorithm 2.3 Est-MDP Goal State Choice

Input: Est-MDP $\mathcal{M}_t^e$, $S_{k,t}^E$, $\mathcal{GP}_t$, current state s_t, g, $k(s,s')$, unexplored set $S_{k,t}^U$
Output: New goal state $s_{goal,t}$, new policy π_t

```
1: $S_{cand} \leftarrow \{s_k \in S_{k,t}^U \mid P_t^{safe}(s_k) > P_{\min}$ and $\Sigma_t(s_k) > \epsilon\}$
2: while $S_{cand} \neq \emptyset$ do
3:     Take next batch of N states $S_N$ with highest uncertainty in $S_{cand}$
4:     for $s_{cand} \in S_N$ do
5:         $p_{reach} \leftarrow Pr^{\max}_{\mathcal{M}_t^e, s_t}(reach_{unsafe,\{(s_{cand},I) \mid I \in S_e\}})$
6:         $p_{return} \leftarrow Pr^{\max}_{\mathcal{M}_t^e, s_{cand}}(reach_{unsafe,\{(s_k,\bar{o}(s_k)) \mid s_k \in S_{k,t}^E\}})$
7:         if $p_{reach} < P_{\min}$ or $p_{return} < P_{\min}$ then
8:             Remove $s_{cand}$ from $S_N$
9:         end if
10:    end for
11:    if $S_N \neq \emptyset$ then
12:        $s_{goal,t} \leftarrow \arg\max_{s \in S_N} \text{score}(s)$
13:        return $s_{goal,t}$, and corresponding policy $\pi_{unsafe,\{(s_{goal,t},I) \mid I \in S_e\}}$
14:    end if
15: end while
16: No goal state identified, end exploration.
```

Again as before, there are three checks carried out to identify a valid goal state among the set of possible candidate states: safety of the state itself, reachability safety and returnability safety.

The state with the highest *score* (goal score function - detailed in Section 2.6.3) that passes all 3 checks is returned as the new goal state. The policy returned from the algorithm is the policy from the solution of the reachability check as this is the correct policy for the agent taking a path from its current state to the goal state.

In line 1, the algorithm restricts the set of candidate goal states to states that are considered safe with high probability, and that we do not already have good knowledge of. Again, if a state s has GP predictive variance $\Sigma_t(s) < \epsilon$ (where the GP is $\mathcal{GP}_t$, given the current data points at time t) then it is not useful to visit according to the safe exploration problem formulation. $P_t^{safe}(s_k)$ is defined as the probability of state s_k being safe according to the current probability distribution $\mathcal{GP}_t(s_k)$.

Reachability and returnability probability calculations are carried out on batches of N states (where N is a user-chosen parameter) with the highest variance and therefore likely highest score. As determining these probabilities requires solving at least one[1] unique constrained reachability optimisation problem for each candidate state, it is computationally expensive to compute scores for too many low-scoring states that are unlikely to be chosen as the next goal.

Both reachability and returnability probability calculations are based on constrained reachability problems over the current Est-MDP with the forbidden set defined as the set of unsafe states according to the safety function, taking the calculated satisfaction probability as the corresponding probability. Again, we assume that the unit of the cost structure of the U-MDP is the time required to carry out the action, but this depends on what exactly the robot system is trying to minimise when it carries out path planning.

Term in constrained reachability problem (Equation 2.4)	Path planning output
$\pi_{F,G}$	Satisfaction policy π
$Pr^{\pi_{F,G}}_{\mathcal{M},\overline{s}}(reach_{F,G})$	Satisfaction probability p_{reach} or p_{return}
$E^{\pi_{F,G}}_{\mathcal{M},\overline{s}}(cumul_{F,G})$	Expected time cost t_s

Table 2.1: Identification of key terms in the constrained reachability problem, and their use in goal identification and task planning.

For the reachability check (line 5), the constrained reachability optimisation problem is set up with the initial state as the current state, and the goal set defined as all states in the Est-MDP where the known component is the candidate state s_{cand}. As well as storing the satisfaction probability, the policy and expected cost of the reachability problem are also stored. The source of these terms is explained in Table 2.1. The expected cost is a variable in the goal state scoring function, and

[1]The returnability constrained reachability problem can actually be solved in parallel for all states. This is because the goal state G (set to the agent's current state) and forbidden states F are the same for each problem, and value iteration solution method [24] simultaneously solves the problem for all initial states.

the policy is stored to be returned if this state is chosen as the next goal. For the returnability check (line 6), the constrained reachability optimisation problem is set up with the initial state as the candidate state s_{cand}, and the goal set defined as the already explored (hence surely safe) states. $\bar{o}(s_k)$ denotes the (possibly noisy) observation associated to s_k.

These probabilities are compared to P_{min} in line 7, which is our defined measure of the level of risk that the agent is willing to accept during exploration. A new goal cannot be identified when the mapping $o(s_k)$ is known within ϵ uncertainty for all reachable unexplored known states, and the exploration algorithm will terminate in line 11 as specified by the problem definition in Section 2.3.2.

2.6.3 Goal Scoring Function

The goal scoring function $\text{score}_t : S_k \to \mathbb{R}$ is designed to indicate how beneficial a state would be to visit and observe, given the knowledge of the Est-MDP at time t. There are many potential approaches to the problem of ranking goal states to visit: different approaches include information theoretic approaches and maximising mutual information between sample points and points of interest.

The scoring function proposed here takes into account the the GP's uncertainty at that state, the expected time cost to reach that state from the current state $t_s = E^{\pi}_{\mathcal{M},\bar{s}}(cumul_{F,G})$ (3), and the reachability/returnability probabilities for the state.

$$\text{score}_t(s_k) = (\Sigma_t(s_k))(t_s)^{-\gamma_1}(p_{reach} \cdot p_{return} - P^2_{\text{min}})^{\gamma_2} \tag{2.24}$$

where $\Sigma_t(s_k)$ is the current GP predictive variance at the known state s_k. The γ values provide relative weightings on different parameters. Setting $\gamma_1 = 1$ results in the first two terms having units of uncertainty (or information) per unit time, which should intuitively be maximised for an exploration task.

The expected time cost t_s can also be adjusted to take into account the measurement time required to take a reading, if the length of time required to observe the environment feature is of the same order as travel times between states. Alternatively, this could be integrated directly into the MDP by specifying a measurement action at each state, with the associated time cost. The LTL constrained reachability formula (Definition 6) would be altered to add the requirement to take a measurement action after arriving at the state.

This would potentially allow for more complex measurement behaviour e.g. probabilistic outcomes for whether the measurement is a success. It would also provide the model solver with better information on the expected cost of taking measurements - which would be useful in the potential extensions to time-bounded exploration discussed in Section 5.2.

2.7 Extension of Estimated MDP Approach to Multi-Step GP Propagation

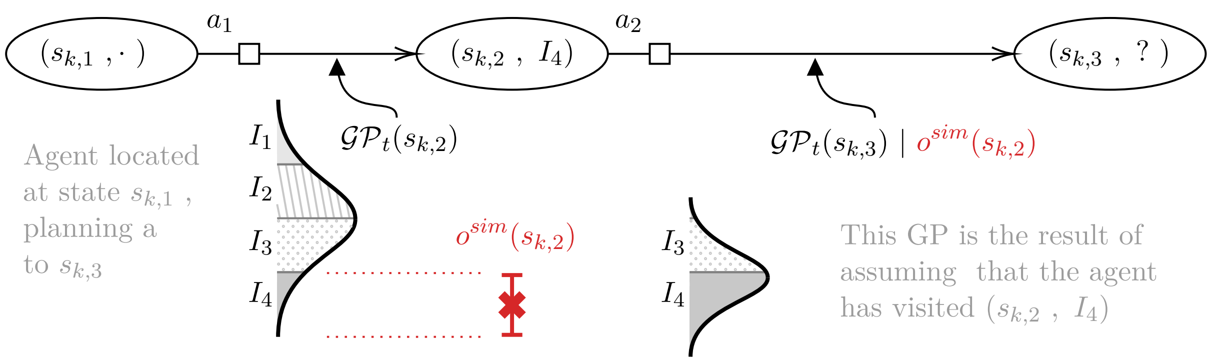

$(s_{k,1}\ ,\cdot\ )$
a_1
$(s_{k,2}\ ,\ I_4)$
a_2
$(s_{k,3}\ ,\ ?\ )$
$\mathcal{GP}_t(s_{k,2})$
$\mathcal{GP}_t(s_{k,3})\ |\ o^{sim}(s_{k,2})$
Agent located at state $s_{k,1}$, planning a to $s_{k,3}$
I_1
I_2
I_3
I_4
$o^{sim}(s_{k,2})$
This GP is the result of assuming that the agent has visited $(s_{k,2}\ ,\ I_4)$

Figure 2.7: Illustration of multi-step path planning where the GP transition function generation is based on GPs that are updated at each step. Using this diagram, the agent is considering a route that starts with a transition to $(s_{k,2}, I_4)$.

Although the Est-MDP construction in Section 2.6.1 allows reachability and returnability calculations to be made based on the agent's current GP model, these calculations are effectively estimates. A more accurate estimate can be produced (at the cost of additional computation time) by taking into account the unknown state component of the agent's state when generating transition probabilities in T_t^e using the GP model (Section 2.6.1).

For the example illustrated in Figure 2.7, when calculating reachability and returnability probabilities the agent should take into account that if it transitions to $(s_{k,2}, I_4)$ then it should condition $\mathcal{GP}_t(s_{k,3})$ on this - which will result in a different distribution (and therefore different transition probabilities) than if it had transitioned to $(s_{k,2}, I_1)$ (which would likely have resulted in assigning much more probability mass to I_1 and I_2, rather than to I_3 and I_4 as it has in the figure).

One method of conditioning the GP predictions on possible unknown state feature values is to add simulated measurements o^{sim} to the GP's dataset for the purposes of planning. This would have to make use of a heteroscedastic GP form (Section 2.1.3) to assign the simulated measurements the correct measurement noise to roughly cover the interval being conditioned on - I_4 in Figure 2.7.

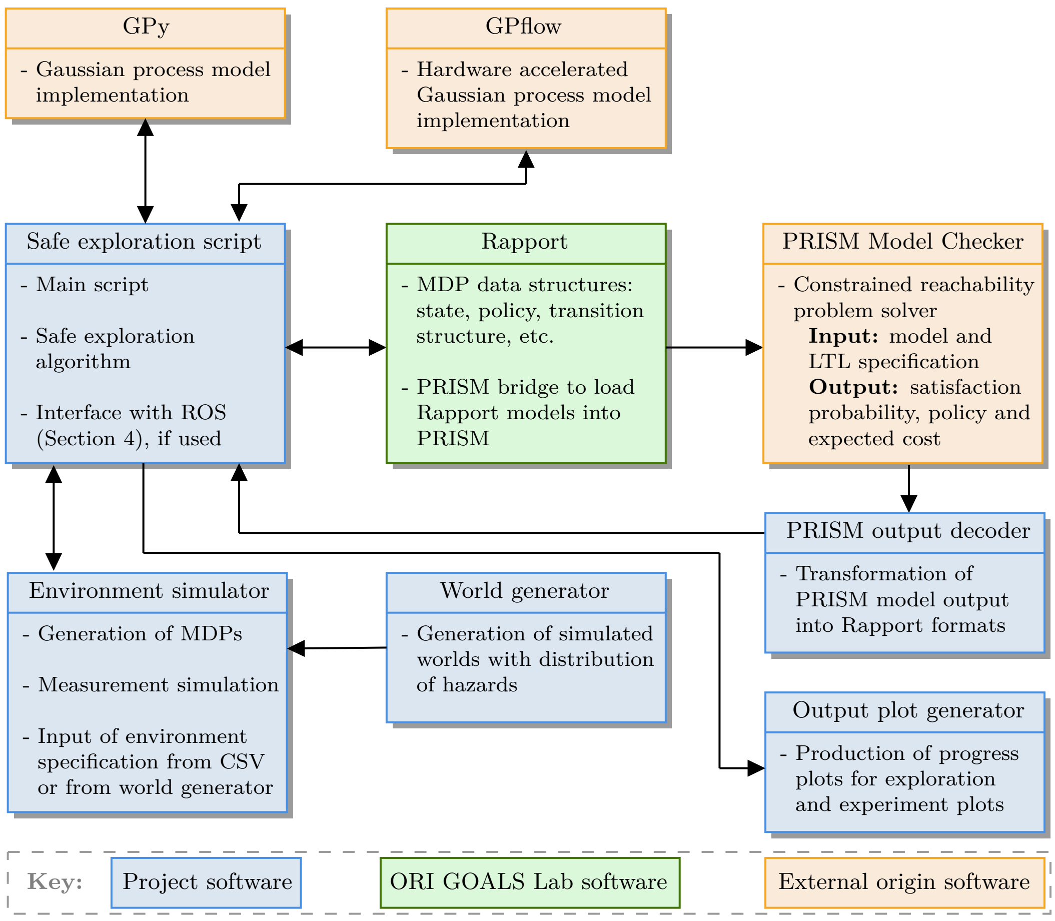
A more elegant form of GP conditioning could possibly be developed based on a non-standard GP classification problem where the labels correspond to Est-MDP intervals.

This could also be extended to condition on more than one transition - which would involve adding more than one simulated observation to the GP dataset. The size of the problem grows exponentially with the number of steps planned ahead in this way.

If the agent is able to take measurements as it moves, then keeping the set of policies generated during the path planning may enable it to dynamically switch to the optimal policy that reflects the ground-truth it is seeing as it travels along the path.

3 Implementation and Experiments

3.1 Software Architecture



GPy
- Gaussian process model implementation
GPflow
- Hardware accelerated Gaussian process model implementation
Safe exploration script
- Main script
- Safe exploration algorithm
- Interface with ROS (Section 4), if used
Rapport
- MDP data structures: state, policy, transition structure, etc.
- PRISM bridge to load Rapport models into PRISM
PRISM Model Checker
- Constrained reachability problem solver
Input: model and LTL specification
Output: satisfaction probability, policy and expected cost
PRISM output decoder
- Transformation of PRISM model output into Rapport formats
Environment simulator
- Generation of MDPs
- Measurement simulation
- Input of environment specification from CSV or from world generator
World generator
- Generation of simulated worlds with distribution of hazards
Output plot generator
- Production of progress plots for exploration and experiment plots
Key:
Project software
ORI GOALS Lab software
External origin software

Figure 3.1: Software architecture diagram, showing the custom software written for the project as well as other tools used.

Figure 3.1 shows software components that make up the implementation. In general, the aim is for the design to be a reusable piece of code that integrates with ORI in-house software. All project code was written in Python, and version control was through Git/GitHub. An implementation in a compiled language such as C++ would increase the performance of the code - this is discussed in Section 3.3.1. Rapport is a probabilistic planning framework that enabled faster prototyping by not having to implement the basics of the standard structures defined in Section 2.1.

Two software libraries were tested for handling GP models, and both fulfilled the GP requirements for the project. Each of GPy [25] and GPflow [26] have different approaches to carrying out GP regression and hyperparameter optimisation. GPflow uses TensorFlow as its back-end engine, which enables acceleration with graphics hardware as opposed to GPy which is purely CPU-based. The differences between the two options are discussed in Section 3.3.1.

Solution of constrained reachability problems was carried out with PRISM [27], a probabilistic model checker written in Java. The PRISM Bridge component in Rapport enabled loading of MDPs into PRISM to be solved, and a custom piece of code transformed PRISM's output back into a usable format.

As well as the two exploration algorithm variants proposed in Sections 2.4 to 2.6, the SafeMDP algorithm [3] was also implemented, to support comparisons between algorithms in the experiments below.

3.2 Evaluation in Simulation

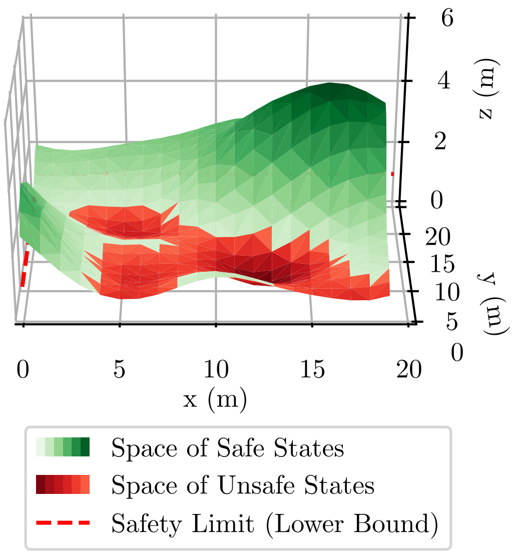
Evaluation in simulation consisted of running the safe exploration algorithms on simulated maps which included a hazard distribution function defined across the map. The known state features of the MDP are x position and y position. For these simulations, a single unknown state feature represents the unknown environmental hazard value, which we place an upper or a lower safety bound on. In MDP form, the map is represented as a 4-connected grid world where the agent has an "up", "down", "left" and "right" action defined at each state (apart from states at the edges). The outcomes of these actions are probabilistic for some experiments below, and deterministic for others.

3.2.1 Programmatically Generated Water Depth Model

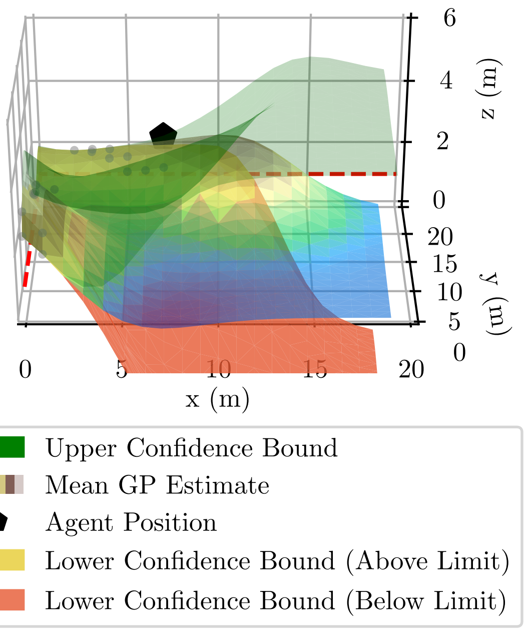
For initial testing of exploration algorithms, environmental hazard functions over the map were generated from a multivariate normal distribution. The covariance matrix of the normal distribution was generated in the same manner as the covariance matrices in Section 2.1.3, using a RBF (2.10) kernel with lengthscale $l = 4.0$ and variance $\sigma = 0.5$. The hazard functions are therefore effectively sampled from a GP prior distribution.

Although the exploration algorithm did not have access to the kernel hyperparameters and was left to optimise those as part of the exploration algorithm, this is not a representative experiment as the environment was drawn from a GP prior distribution and was therefore clearly capable of being modelled by the exploration GP. These tests did however demonstrate working function of the exploration algorithms, and also demonstrated that the U-MDP based safe exploration algorithms outperform SafeMDP by planning new goals to measure that are several steps from the current safe set. This is further discussed in Section 3.2.3.

Figure 3.2a shows an example of a map produced. This is interpreted as an underwater environment where the agent moves around the floor but does not know the depth distribution. It is able to measure



(a) Ground-truth map of the "underwater" environment.



(b) Example exploration progress by the agent.

Figure 3.2: Example plot output of the simulated water depth environment, with a safety specification stating that the agent cannot visit any states below $z = 1.0$. The z position of the floor at each (x, y) coordinate is unknown to the agent.

its z position only at its current location. The safety specification used for this example is $z > 1.0$, and the safe and unsafe areas of the map are coloured according to this specification. The figure shows a small island of safe states around $(x = 10, y = 0)$ - these are however not reachable or returnable from the area that the agent starts off in, so the agent should not pick any of these states as goals to observe.

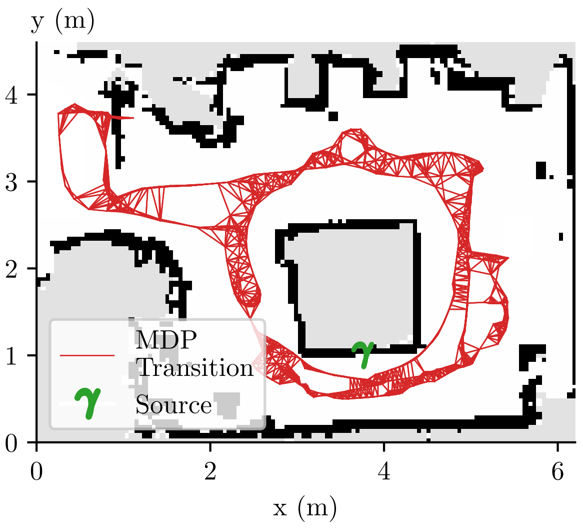
Figure 3.2b shows the agent, currently at $(x = 6, y = 18)$, carrying out an exploration of the map. The transition model is probabilistic: each action has a 0.7 probability of the agent ending up in the intended state (the state one step in the direction of the action), a 0.1 probability each of ending up either side of the intended state (in the orthogonal direction to the action direction), and a 0.1 probability of ending up in its original state. If one of these states is invalid (for example, when attempting to move along the edge of the map) then extra probability is assigned to the agent remaining in the original state.

The agent's current knowledge in the form of a GP is indicated by the bounds and mean estimate shown in the figure. The mean estimate is colour mapped to show height more clearly. The lower bound is shown as either yellow if the agent believes the corresponding state is safe (i.e. if the lower confidence bound function is above the safety bound at that coordinate), or red if the agent believes the state is unsafe. The figure shows the confidence bounds diverging from the mean estimate (representing greater predictive uncertainty) for states that are further away from the set of states the agent has measured. The rate of divergence depends on the (optimised) lengthscale and variance hyperparameters of the GP kernel the agent is using.

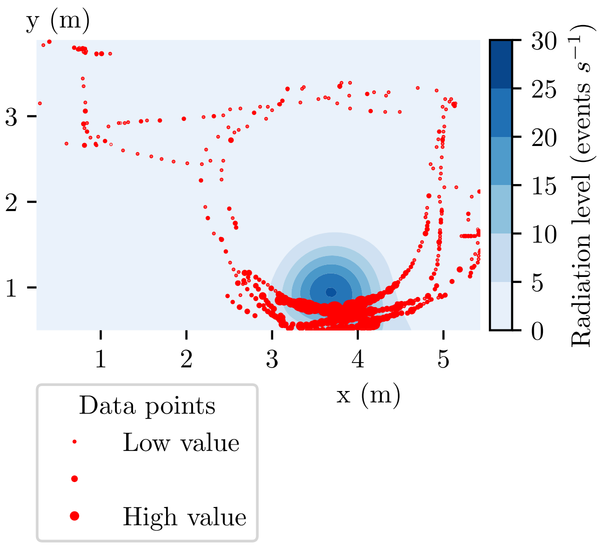
3.2.2 Single Radiation Source Dataset

The experiments in this section, and Section 3.2.3, are designed to validate application of the safe exploration techniques to real-world situations where a mobile robot must investigate a radioactive environment. Given the way we have posed the safety problem, radiation is a good example of a safety hazard to mobile robots: it has the ability to physically damage them (or require them to be disposed of if they become sufficiently radioactive themselves), and it is only measurable in the location the robot is currently in. This dataset was sourced from the Neutron Lab at the University of Lancaster[1] and was captured by a remotely operated vehicle.

The raw data format is a log of Gamma radiation exposure counts and corresponding 2D positions, recorded at constant 1 second timesteps. An MDP was created from this dataset, resulting in a known state space of 554 states and a single unknown state variable to represent radiation counts at an (x, y) coordinate. The transition function defined allows the agent to move to up to 6 closely neighbouring known states. This is shown in Figure 3.3a, with the MDP transitions superimposed onto a LIDAR-created map of the physical environment. Also shown is the approximate location of the Gamma source producing the radiation. For exploration, observations of the radiation level are taken directly from the dataset at the relevant locations.



(a) Physical environment map with superimposed MDP, generated from Gamma radiation dataset to form a relatively densely connected graph.



(b) Predicted posterior distribution over the safety feature, from a GP model trained on the entire dataset (red points).

Figure 3.3: Plots based on Gamma radiation dataset.

Figure 3.3b shows the posterior distribution of a GP model trained on the full dataset. The Matérn 3/2 kernel used gives a good fit for the expected behaviour of a radiation source, as it is symmetrical and falls off in a roughly $\frac{1}{r^2}$ manner as physics would suggest [19].

[1]With thanks to Dr Andrew West at the University of Manchester, for supplying the dataset and physical maps.

An experiment was carried out to prove that the exploration algorithm developed is capable of building a comparable posterior distribution to that of a GP trained on the entire dataset (Figure 3.3b), while only sampling points that are safe. This would verify that it is able to safely produce a useful, predictive model from exploration alone. It is not feasible to directly compare GP models, as they are distributions over an infinite number of functions. Therefore, for the experiment, we characterise the difference between the posterior distribution of the exploration GP and the posterior distribution of the full dataset GP by the Kullback–Leibler (KL) divergence between the two distributions, across $n = 554$ sample points - one at each state. This is calculated as in Section 2.2.1.

We carry out an exploration of the MDP with a safety specification defined as an upper bound on the radiation level, at 25 counts/second. The exploration algorithm is the Estimated MDP approach in Section 2.6, with $P_{min} = 0.95$, batches of $N = 3$ and a Matérn 5/2 kernel used for the GP. The goal scoring function (Section 2.6.3) parameter weightings were set as $\gamma_1 = 1$ and $\gamma_2 = 0.25$. As the approach from Section 2.7 is not used, we simply define two intervals on the radiation level to be expressed in the MDP uncertain state feature, as described in Section 2.3.3. The initial state was the top-left coordinate in the map, corresponding to the first sample point in the dataset. Other than the fixed lengthscale parameter, discussed below, a large uninformative prior distribution was set on the kernel variance hyperparameter to ensure that the agent did not optimise it to zero while it was sampling the low-radiation dataset points near its starting location.

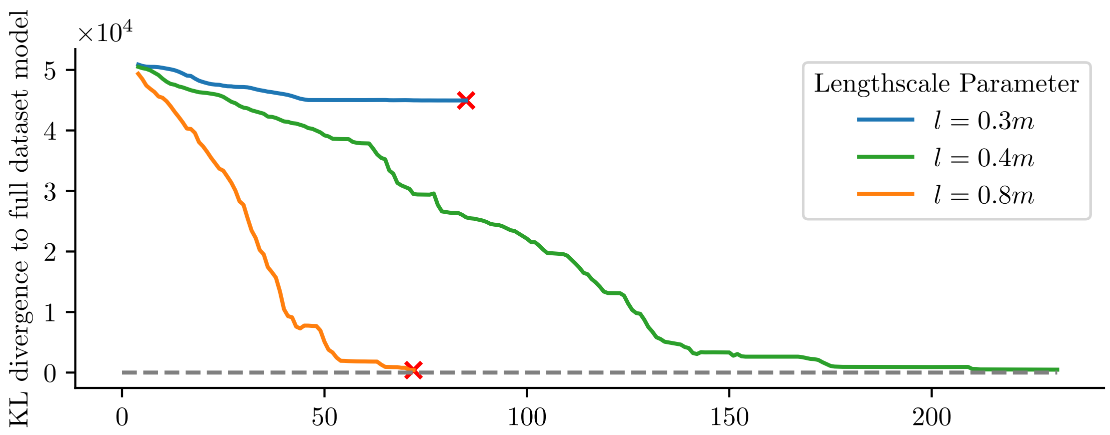


Figure 3.4: KL divergence between the posterior distribution across the states, from iteratively adding safe observations, and the posterior distribution from the full dataset GP model.

Running the proposed exploration algorithm on this dataset shows that it is capable of building a comparable posterior distribution to that of a GP trained on the entire dataset, while only sampling points that are safe. Figure 3.4 shows the result of the experiment, and illustrates the effect of different choices of kernel lengthscale parameter.

A lengthscale of $l = 0.3$m causes the agent to be too cautious, as it terminates after 80 observations believing it has explored the safe set. If the MDP had been created from e.g. a grid map, reducing the grid spacing between states would stop the agent from terminating this early. Here however, we are limited to the MDP structure from the dataset.

On the other hand, a lengthscale of $l = 0.8$m causes the agent to explore rapidly, but it enters an area of high radiation it is not expecting and fails its safety specification. Overall, this demonstrates how the lengthscale hyperparameter defines the upper limit on rate of change of radiation in the environment that the agent should expect.

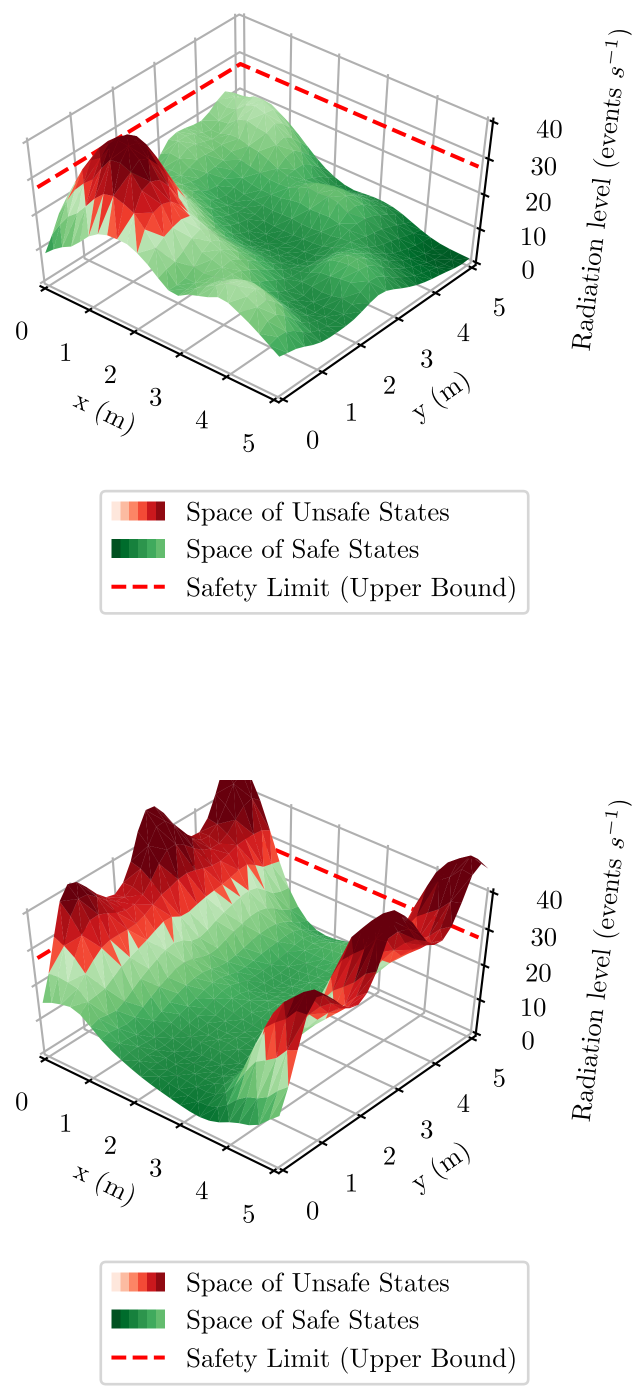
For a reasonable choice of lengthscale e.g. $l = 0.4$m used above, the KL divergence converges towards zero (showing that the agent is building a useful, predictive model) with only ~ 200 of the 554 available observation states observed. It cannot completely converge to zero as the agent is unable to sample from unsafe locations. Within this number of steps the agent also correctly estimates the total number of reachable and returnable safe states to within $\sim 3\%$.

If the lengthscale is assigned an uninformative prior, it tends to result in a shorter lengthscale being chosen as the optimum, as the GP will likely initially try to fit the model to the Poisson noise in radiation counts (for more detail on this see Section 3.3.2). With an uninformative prior assigned, the agent would therefore be on the cautious side, and the lengthscale could be programmatically increased slightly if the agent ran out of states to explore.

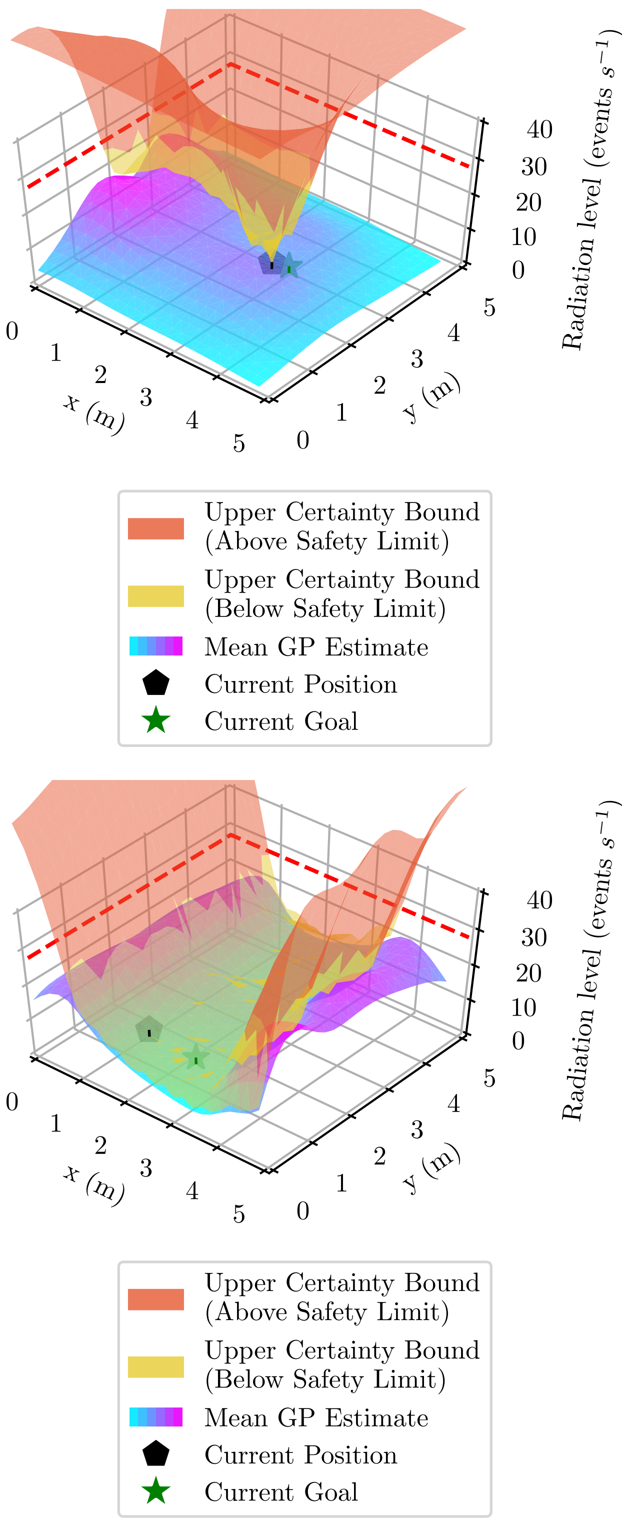
Different kernel functions were compared on their performance for this experiment. The dataset MDP is relatively challenging to explore as the maximum rate of change near the gamma source "spike" is high compared to the physical distance between states in the MDP. The commonly-used RBF kernel (2.10) did not perform well - with a longer lengthscale it became overly certain on predicting low values across the map after exploring the low-count states near the starting location. When assigned a short enough lengthscale it would then switch to modelling the underlying Poisson noise in radiation counts. The Matérn 5/2 kernel (2.11) makes fewer assumptions about smoothness of the underlying function being modelled, and as such was much better able to model the sharp spike in the map while exploring.

3.2.3 Multiple Radiation Source Map

The results of the previous section strongly support the ability of the exploration GP to model the radiation exposure distribution produced by a single low-intensity Gamma source (based on real-world data), in a way that prevents the agent from breaking its safety specification. This section therefore extends the algorithm used above to explore a simulated map containing multiple radiation sources of varying locations and strengths.



(a) Two ground-truth maps of the "radiation" environment, showing which states are safe according to the safety specification.



(b) Example exploration progress by the agent for the corresponding map. The bottom agent has been exploring for longer than the top agent and has made more progress.

Figure 3.5: Example plot output of the simulated radiation environment, where the agent cannot visit any states above 28 events per second. The radiation level at each (x, y) coordinate is unknown to the agent.

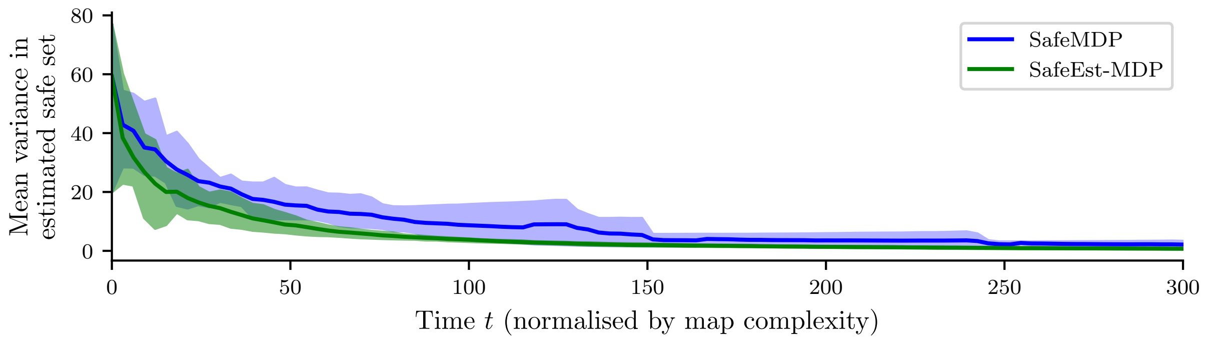
As radiation from multiple sources should be additive (ignoring low-level background radiation, which is small compared to the radiation level from the source and is also captured in the GP's assumption of measurement noise), maps for these experiments are created by a linear combination of affine transformations of the GP-fitted radiation distribution from the single-source dataset in Section 3.2.2. These maps can be generated programmatically but (absent a complex automated map-checking tool) still require manual checking to ensure the map is safely explorable to a reasonable extent and to identify a start location for the agent. Two example maps, and example agent progress exploring them, are shown in Figure 3.5.

Each map is a 5m $\times$ 5m 4-connected grid world where the agent has an "up", "down", "left" and "right" action defined at each state (apart from states at the edges). The map MDP transitions have deterministic outcomes, to allow for comparisons between our exploration algorithm and SafeMDP [3], which does not support probabilistic transition models. With the grid world side length set to 0.2m, this gives a known state space of 625 states. The upper safety limit is set at 28 counts per second. These maps were chosen to be a reasonably representative set of different map types, with the number of sources varying between 1 and 30 and at least half of the states being safe, reachable and returnable.

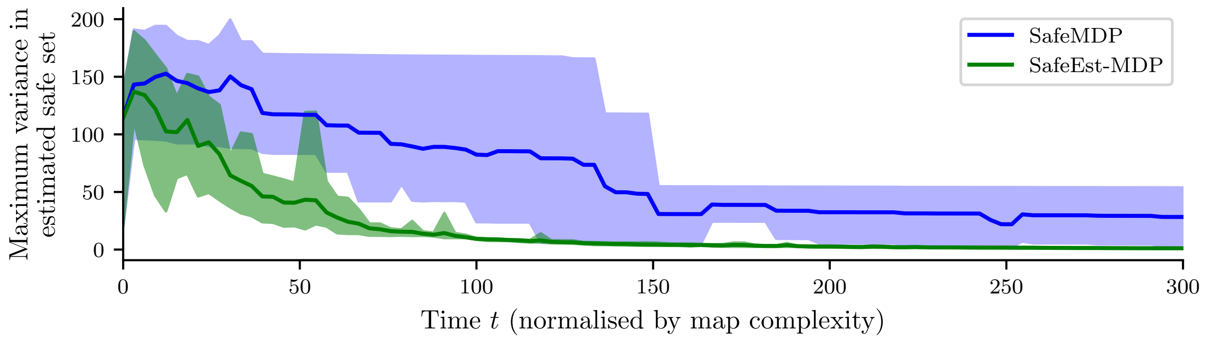
In this section, we refer to our proposed exploration algorithm as defined in Sections 2.4 and 2.6 as SafeEst-MDP. The exploration algorithm parameters and kernel hyperparameters were set up in the same way as in Section 3.2.2. To ensure a fair comparison, the GP kernel was the same for both exploration algorithms and the SafeMDP algorithm implementation included kernel parameter optimisation (Section 2.1.3) as used in SafeEst-MDP - although this was not explicitly specified in [3]. SafeMDP as defined in [3] also not does detail the usage of policies to move the agent around the map, referring to the navigation part of the algorithm as "Safe Dijkstra ... from s_{t-1} to s_t". For implementation simplicity, policies are generated for the SafeMDP implementation using PRISM [27], despite the constrained reachability problem being trivially solvable for a fully-connected explored set with deterministic transitions.

Figure 3.6 shows the results of comparing our exploration algorithm (which we refer to as SafeEst-MDP) with SafeMDP, over a set of 8 simulated maps. The continuous, transparent polygon around the central line for each algorithm represents the performance over all 8 maps for the algorithm (with the top of the polygon representing the maximum value of the y axis parameter over the maps, and vice versa for the bottom of the polygon). The central line represents the mean value across all maps for the algorithm.

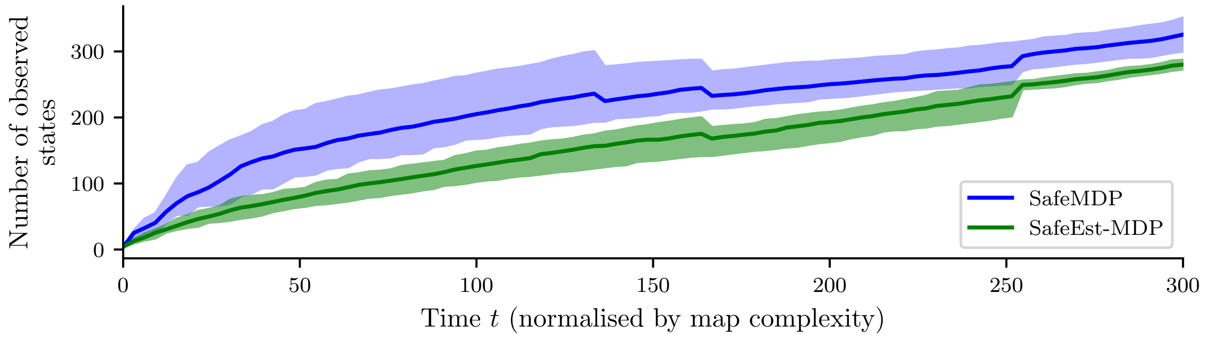
The x axis value for all plots is the (normalised) time value. Although the time cost for each action in the grid world MDP is constant, the maps vary in complexity and safe set size and take differing amounts of time to explore. For easy comparison, the time variable for each map is multiplied by a factor that effectively makes the mean completion time equal for each map, to give representative "error bars" on algorithm performance. The map is considered safely explored when the maximum GP predictive variance across what the agent currently considers the reachable



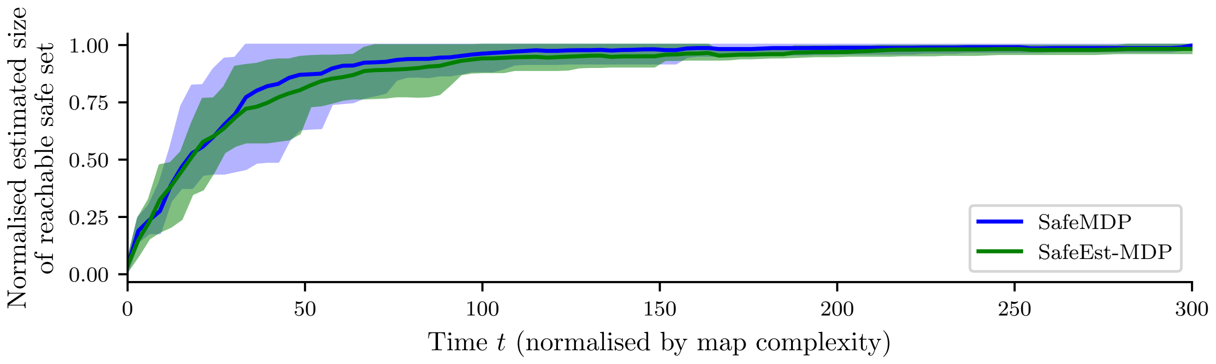
(a) Mean GP predictive variance (across the current estimate of the safe set) vs normalised time for SafeEst-MDP and SafeMDP.



(b) Maximum GP predictive variance (across the current estimate of the safe set) vs normalised time for SafeEst-MDP and SafeMDP.



(c) Cumulative number of states that the agent has taken measurements at vs normalised time for SafeEst-MDP and SafeMDP.



(d) The agent's estimate of the size of the safe, reachable set vs normalised time for SafeEst-MDP and SafeMDP.

Figure 3.6: Comparison plots, on several criteria, between SafeEst-MDP and a SafeMDP baseline. The x axis for each plot is normalised so that it takes roughly the same time to complete, to enable better comparison.

safe set is less than $\epsilon = 10.0$ (corresponding to a standard deviation of 3.3 counts per second), as defined in Section 2.3.2. The mean completion time is defined as the mean of the two times taken by the two algorithms to reach a safely explored state. Note that the algorithms were allowed to keep running past the completion times for Figure 3.6.

Figures 3.6a and 3.6b show plots of the mean and maximum variance across the agent's current determination of the reachable safe set. SafeEst-MDP clearly is much faster at reducing both of these than SafeMDP: on average, over all tested maps, our proposed algorithm is able to determine the radiation level at all reachable safe states (to within ϵ variance) in 52% of the time compared to SafeMDP. This is largely due to SafeEst-MDP's ability to plan routes to goal states that are multiple steps away from the current safe set (as opposed to SafeMDP which only considers neighbours to the current safe set), as well as the use of a significantly more complex goal scoring heuristic. This scoring heuristic disincentivises the agent from taking long routes to observe states with only slightly higher GP predictive variance than much nearer states.

This is especially pronounced for the case of maximum GP predictive variance, where some safe states remain with very high predictive uncertainty even towards the end of exploration where mean predictive variance is converging to zero. This is likely due to SafeMDP's concept of expander states, detailed below.

As well as being more time efficient, Figure 3.6c demonstrates that SafeEst-MDP carries out observations much less frequently than SafeMDP. On average across the maps, it requires 56% fewer observations to reach ϵ maximum GP predictive variance. Many of the differences in exploration speed between the two algorithms would therefore be compounded if a time cost was added to the act of taking observations as well as for moving between states.

Figure 3.6d details the two algorithms' progress as they attempt to determine the size of the reachable safe set. Again, to make the results comparable between maps, the y axis is defined as the proportion of the actual number of safe reachable states for each map, giving 1.0 as the correct determination. For both algorithms the mean line and polygon bounds are monotonic increasing to 1.0 - this demonstrates that both algorithms explore in a safe manner (with our choice of P_{min} and GP kernel parameter priors) and do not determine a state to be safe but later declare it unsafe with more information available.

It is also worth noting that, although the exploration algorithms did not cause the agent to fail the safety specification for any of the tested maps, the Est-MDP formulation results in better robot safety. The bounds-based approaches including SafeMDP do not differentiate between states that are safe with probability $> P_{min}$, whereas the Est-MDP formulation takes into account the exact safety probability of each state, so for example is able to preferentially choose a path along states with e.g. 99% safety probability rather than 95% safety probability.

3.2.4 Expander State Determination

Quickly determining the size of the safe set is explicitly stated as the goal of SafeMDP in [3], and SafeMDP appears to have a slight advantage over SafeEst-MDP at this task as SafeMDP's plot converges marginally more quickly to 1.0. Similarly to the maximum GP predictive variance plot, this is likely due to SafeMDP's use of the concept of expander states. Of the states that are 1-step reachable, returnable, and safe, SafeMDP prioritises any states that are in the *optimistic set of expander states*. This set contains states that, if observed, have the potential to enable the agent to mark more states as safe. This determination is carried out using the Lipschitz constant (Section 2.1.3) of the kernel and underlying safety feature, and is detailed in (3.3). The algorithm will choose the highest predictive variance expander state as its goal if possible, and the highest predictive variance state if not.

$$G_t = \{s \in S_{cand} \mid g_t(s) > 0\} \quad \text{where} \tag{3.1}$$

$$g_t(s) = \left| \{s' \in S_t^U \mid u_t(s) - L \cdot d(s,s') \geqslant \mathcal{LB}\} \right| \quad \text{for a lower bound safety limit } \mathcal{LB} \tag{3.2}$$

$$g_t(s) = \left| \{s' \in S_t^U \mid l_t(s) + L \cdot d(s,s') \leqslant \mathcal{UB}\} \right| \quad \text{for an upper bound safety limit } \mathcal{UB} \tag{3.3}$$

G_t is the optimistic set of expander states, $S_{cand,t}$ is the set of candidate goal states being considered, and S_t^U is the set of unexplored states, all at time t. L is the Lipschitz constant, and $d(.,.)$ is the distance metric defined on S (Section 2.1.3).

The optimistic expander set therefore only contains states for which an optimistic measurement at the state (with "optimistic" meaning at the upper confidence bound function for a lower bound safety limit, or at the lower confidence bound for an upper bound safety limit) would result in at least one state being above / below the safety limit respectively to a high degree of confidence, based on the maximum gradient in the underlying safety feature given by the Lipschitz constant L.

Figure 2.3 contains an illustration of this concept in one dimension - if the red cross near $x = 0$ represents a measurement taken, this measurement has confirmed that any states between $x = -1.9$ and $x = 0.9$ (shown by the green expansion zone line) would certainly be above a lower bound safety limit of -5, given the Lipschitz constant (which, somewhat unrealistically, we know exactly for this example).

These experimental results therefore suggest that the exploration performance of SafeEst-MDP could be improved if it also took into account the concept of expander states. It could potentially do this by adding $g_t(s)$ as another component of the goal scoring function to incentivise the agent to observe states that are likely to result in the largest expansion of the set of states considered to be safe.

However, the Lipschitz constant can be difficult to determine analytically, and extending this concept to handle more complex safety specifications than an upper or lower limit may be challenging.

3.3 Further Discussion

Exploration time is of course slightly reduced for a map with deterministic transitions (when using our proposed algorithm) as the agent can plan shortest paths and can determine the exact time it will take to reach a state. In comparison, with a probabilistic transition model as defined in Section 3.2.1, the policy generated from the constrained reachability problem will choose paths with a "safety buffer" from the edge of any potentially unsafe areas, so even if an action accidentally takes it closer to the unsafe area it will be able to recover.

3.3.1 Implementation Performance Evaluation

The comparisons in this section are rough and not completely informative as they represent only one size of map with deterministic transitions - larger maps (and probabilistic transition outcomes) will scale the size of the problem and required computational effort. It is worth noting that problem complexity does not necessarily scale exponentially with the total map size in terms of number of known state feature values - in most instances only a small unexplored area around the robot is safe enough to plan paths to. It will however scale with the complexity of the MDP transition structure and the number of discretisation intervals (Section 2.3.3).

In the map examples tested, the SafeEst-MDP computation time for choosing goals is roughly double the computation time for SafeMDP. For the sizes of MDP specified by the maps, SafeEst-MDP takes an average of ~ 5 seconds to choose a new goal state (2019 generation 6-core i7 processor, 16GB RAM). For SafeEst-MDP this is split into the time required to update and predict with the GP model ($\sim$5%), the time required to build the Est-MDP model and parse data formats ($\sim$45%) and the time required for PRISM to solve constrained reachability problems ($\sim$50%).

The largest bottleneck in speeding up the time required to choose a goal is the constrained model solution using PRISM [27] - this is already an optimised piece of software so it is unlikely (without significant effort) that it can be made to solve the problems considerably faster.

There was a negligible performance difference between GPy [25] and GPflow [26] - for the size of the maps used, any performance benefit from running the GP regression in parallel was counteracted by the time taken to load the model into the GPU hardware. It may be that for more complicated maps, or GP models with higher dimensionality, GPflow will offer an overall performance benefit over GPy.

The largest potential improvement in computational performance would therefore likely come from optimising the algorithm itself to require fewer PRISM runs to explore, and from optimising the logic that generates the Est-MDP structure and implements the SafeEst-MDP algorithm. Significant performance gains could likely be achieved by rewriting this logic in a compiled language rather than Python.

3.3.2 Modelling of Radiation Environments with Gaussian Processes

Sections 3.2.2 and 3.2.3 demonstrate the ability of our safe exploration methods to tackle environments with relatively low radiation levels (the maximum count per second (cps) rate in the radiation dataset is 38) using a standard GP regression. This is in common with previous literature, where similar GP models have successfully been used to map radiation fields at low exposure rates [28]. Due to the statistical nature of radiation exposure, a more complex analysis is required to be able to handle radiation levels that are high enough to conceivably harm the robot.

As detailed in Section 2.2.2, radiation exposure is governed by the Poisson distribution, specified by the mean exposure rate parameter λ (2.17). In this case the safety function o modelled by our GP is the spatial distribution of λ, i.e. $\lambda(\mathbf{x})$, which we can take noisy measurements of by counting the number of detected radioactive events $y(\mathbf{x}) \sim \text{Pois}(\lambda(\mathbf{x}))$.

In Definition 7 we suggest a Gaussian distribution as the GP likelihood function $p(y|\lambda) = \mathcal{N}(\lambda, \sigma_n^2)$. However, here the correct likelihood function for this situation is a Poisson distribution $p(y|\lambda) = \text{Pois}(\lambda)$. At low values of λ, the Gaussian distribution shape differs considerably from a Poisson distribution, and also assigns some probability to $\lambda < 0$, which is clearly non-physical.

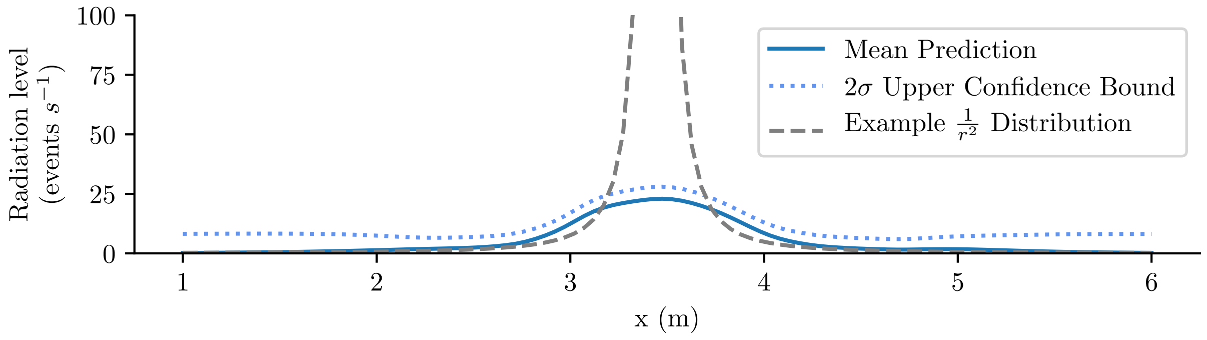
At higher rates[2], the normal approximation to a Poisson distribution (2.18) is a very good approximation and the likelihood function therefore converges to a Gaussian. However, the variance of this Gaussian distribution is also λ: as the mean exposure rate increases, the standard deviation increases as the square root of the mean exposure rate. It is of course possible to reduce uncertainty by sampling for n time periods instead of 1, causing the standard deviation to reduce as $\sigma = \frac{\lambda}{\sqrt{\text{n}}}$ according to the central limit theorem. This is not a practical solution as it requires pausing the robot for long periods in areas where it will accumulate more damage from the radiation it is measuring - and the $\sqrt{n}$ term leads to diminishing returns on reducing uncertainty.

To tackle this problem, as in [29], the GP is used to model the logarithm of the exposure rate $\log(\lambda(\mathbf{x}))$ by combining the GP with a logarithmic link function along with a Poisson likelihood function. This also limits $\lambda(\mathbf{x})$ to positive values only, as desired. Without a Gaussian likelihood, the measurement noise parameter (Section 2.1.3) is removed, reducing the number of hyperparameters to be optimised by the GP regression. However, the new GP model is non-conjugate and the GP regression predictive equations (2.7 to 2.9) no longer apply: the regression must be carried out with approximate methods.

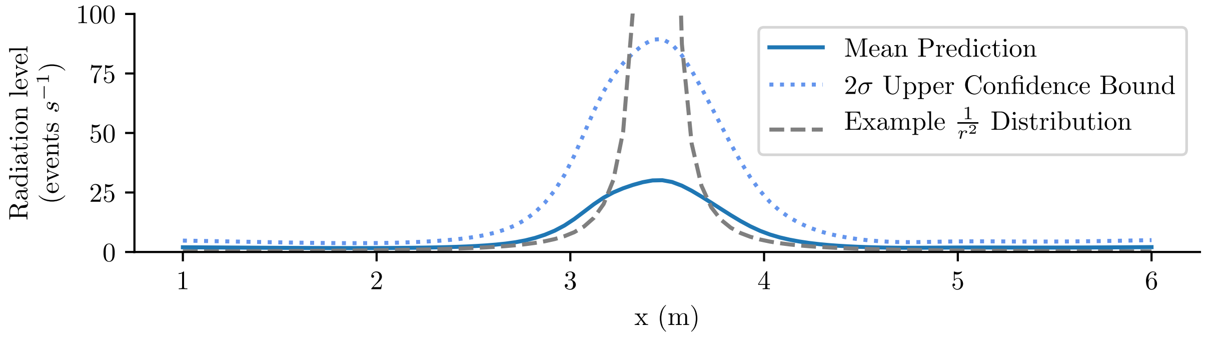
These include Laplace approximation (implemented by GPy), sampling-based methods such as Monte Carlo Markov Chain (MCMC, implemented by GPflow), variational inference approaches and expectation propagation [4, 30, 31]. With a Poisson likelihood, the GP posterior distribution

[2]Rough calculations indicate that 10Gy/hr exposure (which is at the lower end of the radiation exposure rate that AVEXIS (Figure 1.1a) is designed to tolerate) would correspond to 1300 cps with a TF Scientific RadEye survey meter.

is reasonably well characterized by its mode so the Laplace approximation is reasonable [30]. The theoretical limit of MCMC with an infinite number of samples gives the true GP posterior distribution, but sampling methods clearly require much more computation than an equivalent method based on approximation using other distributions. In both the Laplace approximation and MCMC cases, the mean and confidence bound predictions would have to be determined by sampling from the GP posterior. As expected, all of this significantly increases the computational effort required to carry out predictions with a Poisson likelihood GP compared to the standard GP model.



(a) Single axis posterior distribution produced with a Gaussian likelihood GP. The confidence bounds are symmetrical about the mean prediction line, meaning that the GP is assigning some probability to negative λ.



(b) Single axis posterior distribution produced with a Poisson likelihood GP, computed using the Laplace approximation implementation in GPy. The lower certainty bound is much closer to the mean prediction than the upper (and is restricted to being > 0) but is not shown here for clarity of the figure.

Figure 3.7: Comparisons of the two likelihood functions using the radiation dataset, shown along the x axis with a fixed y value to take the plot through the predicted peak location of the radiation level distribution.

Although the standard GP regression of the radiation dataset produced a reasonable mean function for the purposes of multiple source radiation maps in Section 3.2.3, it is clear from the confidence bound plot in Figure 3.7a that it is not well modelling the underlying behaviour. It has reasonably high predictive uncertainty across most low radiation exposure states, and is overconfident in its predictions around the radiation peak - where the predictive uncertainty actually decreases compared to the low radiation exposure states.

The Poisson likelihood GP fit in Figure 3.7b is clearly significantly better - it correctly assigns most of the plane to ~ 0 value background radiation with high confidence, and much more cleanly fits a possible underlying $1/(r^2)$ function inside its error bounds, showing much higher uncertainty around the peak. The uncertainty increases significantly as the number of radiation events increases, as expected from the underlying form of the normal approximation to the Poisson distribution. The Laplace approximation method to produce the plot was verified by also running an MCMC equivalent with GPflow, which gave a high degree of agreement on the mean function and confidence bounds.

This section (in particular, Figure 3.7b) highlights that the radiation safety constrained exploration problem is particularly difficult because of the widening uncertainty over λ at higher exposure rates. This makes it very difficult for the agent to reduce its confidence bounds close to the safety limit, even with repeated sampling.

A good compromise for highly radioactive environments may be to employ a heteroscedastic GP model with a standard Gaussian likelihood. This allows for manually increasing the noise variance assigned to higher value observations. Although the heteroscedastic noise model still excludes a closed-form solution, it may prove to be an easier model to optimise hyperparameters over than a Poisson likelihood GP. The model would not work well for areas with low count values where the normal distribution is not a good approximation to the Poisson distribution, but this is a reasonable trade-off as these low-intensity areas are clearly the furthest from any safety limit.

3.3.3 Additional Gaussian Process Model Considerations

Another real-world consideration for applying safe exploration techniques to a highly radioactive environment is the interaction of radiation with physical environmental features. Although we assume for this report that the distribution of radiation is independent of physical features, a real-world environment with high levels of radiation will exhibit behaviour such as the physical geometry attenuating, reflecting or completely blocking radiation.

This is a problem for our modelling as it can effectively introduce discontinuities in radiation level. For example, corners are particularly "dangerous": a mobile radiation survey robot may detect very little radiation as it moves along a thick concrete wall, until it moves just around a corner where it now has direct line-of-sight to a high activity radiation source which gives it a "lethal" dose of radiation.

Our suggested method for handling this problem is to alter the GP's prior distribution by using a non-zero mean function, the case covered in (2.9). Assigning a higher mean radiation level value $\mathbf{m}(X)$ to certain geometrical features in the mobile robot's map (such as the corners of thick walls) would make the agent more wary of these features. This should encourage it to approach the "dangerous" geometry from further away, ideally where it has direct line-of-sight to whatever is around the corner, but from a safe distance. As it approaches it would then see an upwards gradient in radiation exposure rather than the step change of moving around the corner.

4 System Integration with ROS

4.1 Software Architecture

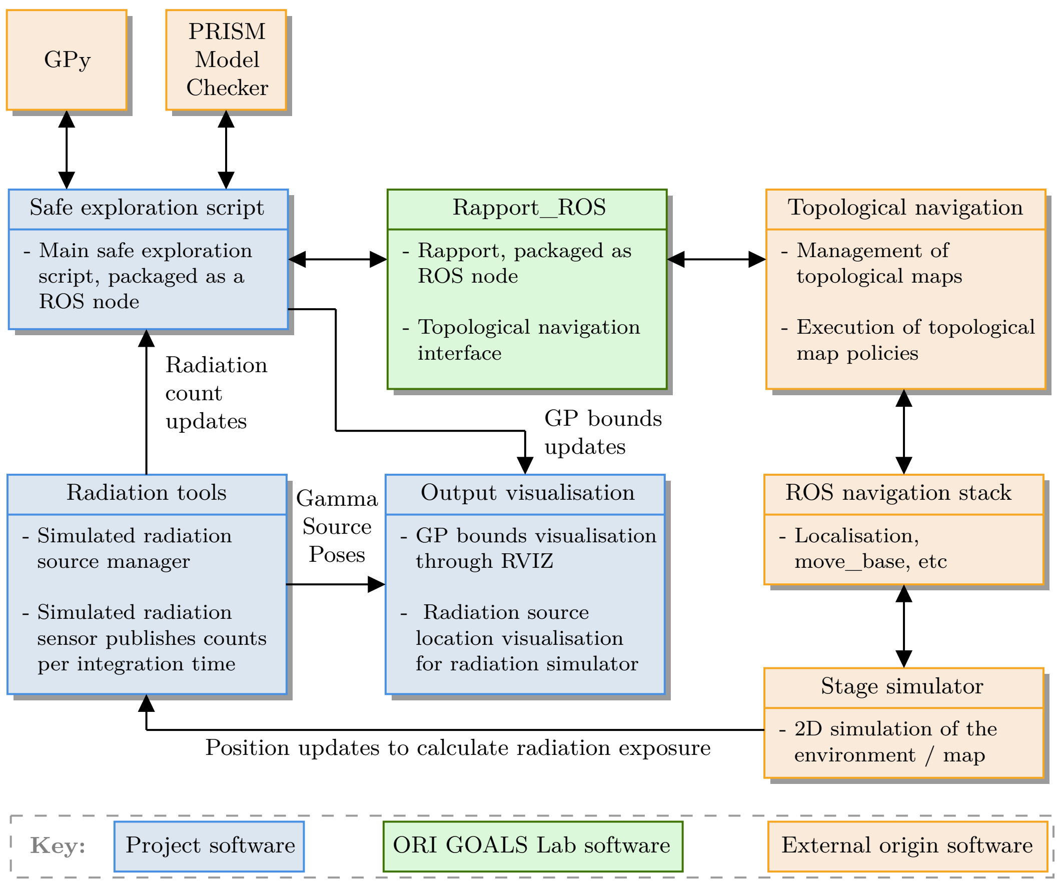

GPy
PRISM Model Checker
Safe exploration script
- Main safe exploration script, packaged as a ROS node
Rapport_ROS
- Rapport, packaged as ROS node
- Topological navigation interface
Topological navigation
- Management of topological maps
- Execution of topological map policies
Radiation count updates
GP bounds updates
Radiation tools
- Simulated radiation source manager
- Simulated radiation sensor publishes counts per integration time
Gamma Source Poses
Output visualisation
- GP bounds visualisation through RVIZ
- Radiation source location visualisation for radiation simulator
ROS navigation stack
- Localisation, move_base, etc
Stage simulator
- 2D simulation of the environment / map
Position updates to calculate radiation exposure
Key:
Project software
ORI GOALS Lab software
External origin software

Figure 4.1: Software architecture diagram for the ROS-integrated system, showing the custom software written for the project as well as other tools used. Some important flows of information are labelled on the corresponding arrow showing communication between system components.

This section is designed to demonstrate how a full-stack robotic system would employ the safe exploration code to carry out a real-world safe exploration task. The implementation from Section 3 was integrated into a robotic framework called the Robot Operating System (ROS) [32], to interface with other components frequently used in ORI robotic systems. A basic overview of the components running under ROS is shown in Figure 4.1. To allow the system to be used as a technology demonstration, work was also carried out to produce a real-time graphical system for displaying the radiation environment around the robot and its current GP model.

Although this system was originally designed to be run on a Clearpath Robotics “Jackal” mobile robot for a demonstration in the ORI, this was replaced by the Stage simulation package towards the end of the project. Stage provides the simulation of the mobile robot, its sensors and its physical environment. This would be replaced by the actual robot and navigation sensor interfaces for a real trial.

4.1.1 Topological Navigation Stack

As opposed to the simulated experiments in Section 3, with a real mobile robot we do not simply have to navigate across an idealised grid map. The problem of localising the robot within a physical environment and navigating around it is handled with the standard ROS navigation stack. This carries out sensor fusion between the input data from the robot LIDAR sensor and the dead reckoning of the robot’s motor control.

For reliable higher-level navigation functionality, we make use of the topological navigation framework from the STRANDS project [33]. The STRANDS topological navigation framework provides the concept of a *topological map*, where the physical map is covered with a graph made up of topological *nodes*, with transitions between nodes represented by the edges of the graph.

For carrying out navigation between nodes, the standard ROS move_base navigation component is used. The topological navigation system will attempt to guide the robot to the centre of the topological node specified by its current progress along the path (equivalent to a policy) that it is following.

Rapport_ROS provides functionality for transforming a topological map into an MDP. The high-level approach to carrying out safe exploration is therefore to programmatically generate a grid-shaped topological map (either 4-connected or 8-connected) over the space to be explored (using a script in the safe exploration package), which is then transformed into an MDP by Rapport_ROS. 8-connected grid maps provide smoother movement for robot navigation, but increase the planning load by a large factor due to the much larger action space. There is no particular requirement to use a square grid map, and a set of topological nodes could be laid out in a hexagonal pattern for example.

4.1.2 Safe Exploration Package

This ROS package encapsulates the safe exploration code documented in Chapter 3. Given an MDP produced from the topological map and a starting set of observations, it manages the progress of the safe exploration algorithm (Section 2.6) in an asynchronous manner. When policies and goals are generated by the safe exploration script, these are converted into topological navigation paths between nodes and passed to the topological navigation stack. The exploration algorithm then blocks on the navigation progress until the goal state is reached. Measurements are carried out by pausing at a topological node and inputting a reading from the simulated radiation sensor node.

4.1.3 Radiation Tools Package

In the form of a ROS package to allow easy integration with the system, this software package is designed to allow arbitrary placement of radiation sources around a map (either in simulation or for a physical robot test - a real robot will “detect” the simulated radiation in the same manner).

Also provided is a simulated radiation event counter that measures the robot’s exposure to radiation. This was designed to use a standard radiation sensor message format, to make it a drop-in replacement for a physical radiation sensor - either a real one or a physical simulation of radiation. The mean radiation count rate at an instant in time $\lambda(\mathbf{x})$ is calculated based on the list of n radiation sources with locations specified by $\mathbf{x}_i^{src}$ and strengths specified by s_i^{src}:

$$\lambda(\mathbf{x}) = \sum_i^n \frac{s_i^{src}}{\|\mathbf{x} - \mathbf{x}_i^{src}\|^2} \tag{4.1}$$

To allow for calculation of radiation exposure while the robot is moving, the radiation sensor node integrates $\lambda(\mathbf{x})$ over time to average it, and then samples a value to report as the total counts during the integration period according to (2.17). Simple adjustment of the mean rate λ (which is taken to refer to events per second in this report) is required if the counter integration period is set at anything other than one second.

The radiation tools are also capable of independently logging the maximum radiation exposure rate and total accumulated radiation dose, to provide an independent benchmark for characterising exploration performance.

Two approaches have been discussed for a physical simulation of radiation exposure, for use in an exploration demonstration on a real robot. A light-based system would use visible light of a specific wavelength to simulate radiation, with light sensors on the robot measuring the exposure. Alternatively, the ORI has access to a physical radiation simulation system developed at the University of Manchester which uses variable-power high frequency radios as “radiation sources” and a radio signal strength sensor as the “radiation exposure” sensor. Either of these physical radiation simulators could be set up to be computer controllable, which would allow for the possibility of simulating a hazard distribution that is not stationary in time. This would enable testing of a spatio-temporal GP extension of the current exploration GP modelling approach, as in [9].

4.1.4 Output Visualisation Package

To add to the system’s utility as a technology demonstration, a tool was created to output the agent’s GP model in real-time as a visualisation in RVIZ, the ROS 3D visualisation tool. This was the work of Michal Staniaszek at the ORI, and makes use of RVIZ point clouds to show the current GP mean prediction and upper confidence bound across the physical environment map.

As shown in Figure 4.3, the colour of the confidence bound point cloud points is assigned based on whether the agent believes the corresponding area is safe according to the safety specification - in a similar way as the plots in this report (Figures 3.2b and 3.5b).

4.2 System Validation Outputs

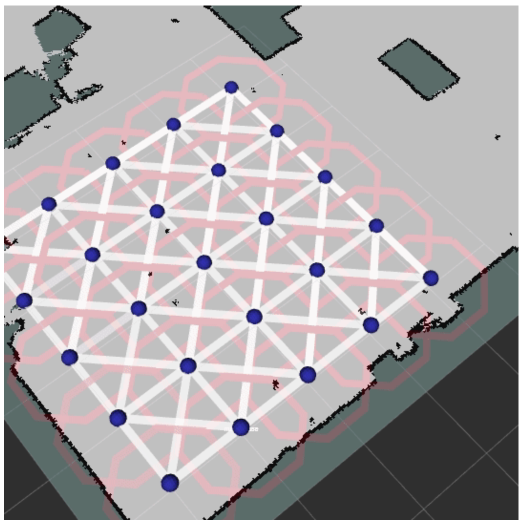


Figure 4.2: An 8-connected topological map laid out over a physical collision map, with the map nodes shown by blue circles.

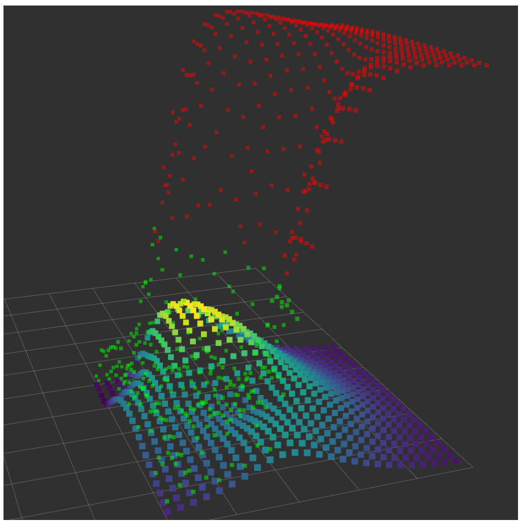


Figure 4.3: An Example GP model mean and upper bound point cloud visualisation in RVIZ.

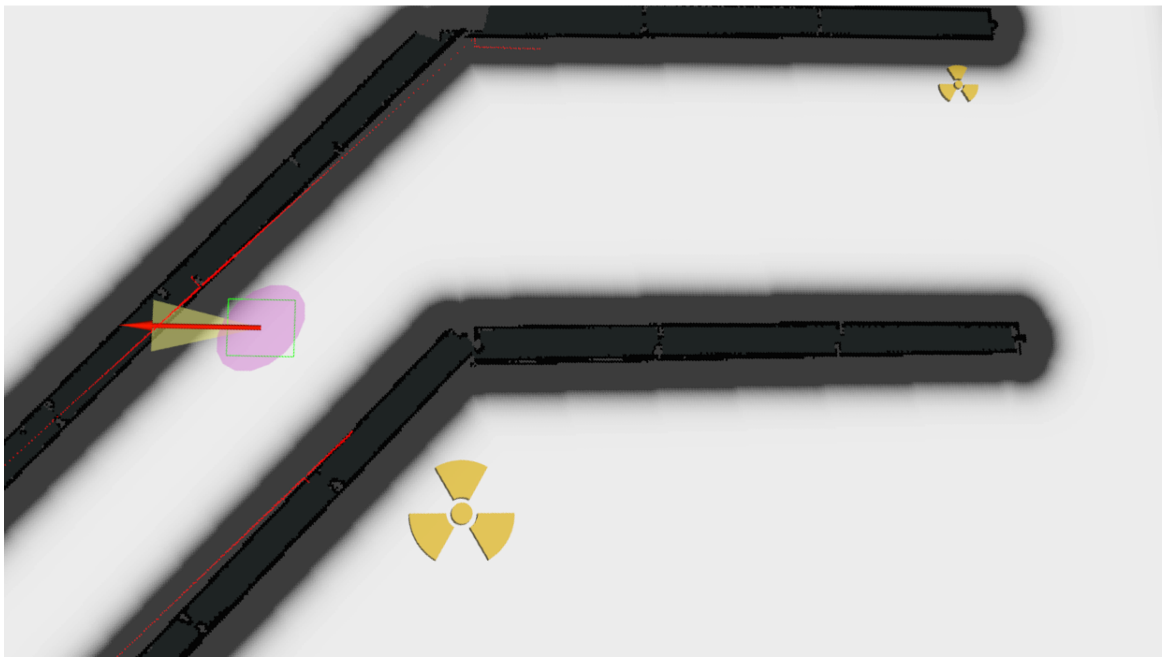


Figure 4.4: RVIZ output showing a simulated mobile robot in a corridor, where the black lines are walls that it has recorded in its map. The pink ellipse represents the robot's belief about its location, based on the LIDAR input (shown as red lines on the insides of the walls). Also shown are markers for two radiation sources of differing strengths, added by the radiation tools package. The radiation detector simulation is also simulating radiation event counts on the robot.

These figures show correct operation of several of the components described in Section 4.1.

5 Conclusions

5.1 Summary of Contributions

In the course of this project, we have investigated the theory behind safe exploration in MDPs, developed new formalisms and algorithms, and evaluated their effectiveness in representative safety constrained tasks based on simulation using real-world data.

The new formalisms developed during this project show significant promise based on the experimental results collected, and allow for more expressive safety specifications than previous approaches. The SafeEst-MDP algorithm, which is based on the new formalism, generally outperforms previous safe exploration algorithms in several measures as shown by a set of experiments run on simulated maps. The increased complexity of our algorithms means that planning processing time is significantly increased compared to simpler exploration algorithms such as SafeMDP [3]. However, the experiments demonstrate that planning is still acceptably fast for reasonable problem sizes, and for many applications the heavy computation work can be performed remotely from the mobile robot.

This report also includes a more in-depth analysis and implementation of kernel choice and hyperparameter optimisation than existing safe exploration literature, which tend to leave this area relatively unexplored. For the radioactive environment application case, an in-depth analysis was carried out using a Poisson likelihood function in Section 3.3.2, which demonstrated the need for careful choice of the GP model type used to model highly radioactive environments, where uncertainty in radiation level is a significant concern. The difficulty of the radiation environment exploration task that we set for the agent demonstrates that it would also work well on comparatively simpler tasks where a Gaussian likelihood is more appropriate.

Other than the theory developed, several software tools were created to enable work on safe exploration and GP modelling. These tools are designed with future use and extension in mind, as they integrate with existing ORI code. As well as the core data structure and algorithm implementations, work was also carried out to integrate the safe exploration codebase into a full robotic system in Section 4, requiring it to cooperate with separate mapping, localisation and local navigation components also running on the robot. This also included the construction of additional tools for simulating a radiation environment in a real-world map, and expressively visualising the robot's model of its environment. This all enables the feasibility of a full safe exploration robotic system to be demonstrated in simulation, with a view to run the demonstration on a real-world mobile robot in the future.

5.2 Potential Future Work

There is significant potential for future work to be carried out, both in terms of improving the underlying exploration approach and extending it to tackle new exploration tasks.

Included in our formulation of the Est-MDP is the potential for unknown state features to affect the transition model - both in terms of action costs and transition probabilities. It is worth investigating how the safe exploration algorithm behaves for a case where the transition model structure does depend on unknown states, potentially in an underwater vehicle setting where currents may affect the robot's ability to navigate. This concept was not explored in-depth for this report, particularly as the radiation level is assumed not to affect the robot's ability to navigate in the experimental sections.

Integrating GP predictions into the cost structure would be one approach to problems where there is a cumulative exposure limit as well as an upper bound limit on the hazard - this is particularly applicable to a radioactive environment. This could lead to a more complex and interesting exploration task, such as requiring a mobile robot to explore as much of a radioactive environment as possible without surpassing a cumulative radiation exposure limit. The ability to specify these tasks fully in co-safe LTL would be elegant and highly useful, and would require more work.

Extending our exploration approach to carry out safe exploration with Bayesian optimisation or goal-driven exploration tasks (such as those discussed in Section 1.2) would be other clear examples of future work. Previous literature provides a starting point as to how this has been done with the SafeMDP [3] algorithm.

Additional investigation of the POMDP-based methods described in Section 1.2 would enable better comparisons between their underlying models and ours, and could show how a POMDP representation might be transformed into an Est-MDP representation and vice versa. Along the same vein, the information theoretic Bayesian optimisation theory in this field of literature could be used to develop better exploration algorithms. The current goal scoring function in Section 2.6.3 depends largely on the GP predictive uncertainty to identify which states are useful to measure - this is generally considered a myopic approach to maximising information from exploration. The goal scoring function could also be improved by the addition of the concept of expander states from SafeMDP, as described in Section 3.2.4.

In terms of the specific application of safe exploration to radioactive environments, more investigation could be carried out into how best to include prior information on risky physical geometry (such as the corners of thick walls) using the GP prior mean function, as discussed in Section 3.3.3. Further development and testing of this concept would require much more complex models of radiation behaviour than this project had access to, and possibly real-world experimentation.

The initial investigation into modelling radiation with a Poisson likelihood GP in Section 3.3.2 suggests that a Laplace approximation method works well for our gamma radiation dataset, but more work could be carried out to compare the efficiency and accuracy of different approaches such as expectation propagation. As the Poisson likelihood GP requires significantly more computational effort than the GP models used elsewhere in this report, it may also be worth looking into sparse approximations [4] for this GP model.

Another point for future investigation would be to make better use of continuous counting measurements e.g. from a radiation event sensor. This sensor would be continually detecting events as the robot is navigating in the radioactive environment example, but at present we only make use of readings that we take when we have arrived at states of interest. Continuous monitoring of radiation level could also feed into the policy safety check that runs as the robot is navigating. For example, if the robot detects significantly higher radiation than it expected when it planned its path, it may be a good idea to stop and reconsider.

In terms of implementation and integration, it would be interesting to apply the safe exploration system to a situation where the robot has incomplete knowledge of the physical environment as well, which it may handle using e.g. simultaneous localization and mapping (SLAM) techniques. The exploration algorithm could be tweaked to encourage the agent to visit states that may increase knowledge of the physical environment as well as knowledge of the unknown hazard distribution.

Finally, a full implementation of the multi-step GP prediction propagation (discussed in Section 2.7) would be useful to characterise the trade-off between more accurate estimation of the safety of paths and additional computational load.

Bibliography

[1] E. Ackerman, “Unlucky robot gets stranded inside Fukushima nuclear reactor, sends back critical data,” accessed 2020-04-2. Available: https://spectrum.ieee.org/automaton/robotics/industrial-robots/robot-stranded-inside-fukushima-nuclear-reactor

[2] L. Chang, “The robots Japan enlisted to clean up Fukushima have been destroyed from radiation exposure,” 2016, Business Insider, accessed 2020-04-25. Available: https://www.businessinsider.com/fukushima-robots-destroyed-by-radiation-2016-3

[3] M. Turchetta, F. Berkenkamp, and A. Krause, “Safe exploration in finite Markov decision processes with Gaussian processes,” in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, p. 4312–4320.

[4] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning.* MIT Press, 2006.

[5] J. García, Fern, and o Fernández, “A comprehensive survey on safe reinforcement learning,” *Journal of Machine Learning Research*, vol. 16, no. 42, pp. 1437–1480, 2015. Available: http://jmlr.org/papers/v16/garcia15a.html

[6] Y. Sui, A. Gotovos, J. W. Burdick, and A. Krause, “Safe exploration for optimization with Gaussian processes,” in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37.* JMLR.org, 2015, p. 997–1005.

[7] T. M. Moldovan and P. Abbeel, “Safe exploration in Markov decision processes,” in *Proceedings of the 29th International Conference on Machine Learning*, 2012, pp. 1451–1458.

[8] A. Wachi, Y. Sui, Y. Yue, and M. Ono, “Safe exploration and optimization of constrained MDPs using Gaussian processes,” in *AAAI Conference on Artificial Intelligence*, 2018.

[9] A. Wachi, H. Kajino, and A. Munawar, “Safe exploration in Markov decision processes with time-variant safety using spatio-temporal Gaussian process,” *arXiv preprint arXiv:1809.04232*, vol. abs/1809.04232, 2018. Available: http://arxiv.org/abs/1809.04232

[10] M. Turchetta, F. Berkenkamp, and A. Krause, “Safe exploration for interactive machine learning,” in *Advances in Neural Information Processing Systems*, 2019, pp. 2887–2897.

[11] R. Marchant, F. Ramos, and S. Sanner, “Sequential bayesian optimisation for spatial-temporal monitoring,” in *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, ser. UAI’14. Arlington, Virginia, USA: AUAI Press, 2014, p. 553–562.

[12] G. Flaspohler, V. Preston, A. P. M. Michel, Y. Girdhar, and N. Roy, “Information-guided robotic maximum seek-and-sample in partially observable continuous environments,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3782–3789, 2019.

[13] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, “Learning-based model predictive control for safe exploration,” in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 6059–6066.

[14] K. Polymenakos, L. Laurenti, A. Patane, J.-P. Calliess, L. Cardelli, M. Kwiatkowska, A. Abate, and S. Roberts, “Safety guarantees for planning based on iterative Gaussian processes,” 2019.

[15] Mausam and A. Kolobov, *Planning with Markov Decision Processes: An AI Perspective.* Morgan & Claypool Publishers, 2012.

[16] A. Pnueli, “The temporal semantics of concurrent programs,” *Theoretical Computer Science*, vol. 13, pp. 45–60, 1981.

[17] F. Teichteil-Königsbuch, “Stochastic safest and shortest path problems,” in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012, p. 1825–1831.

[18] B. Lacerda, F. Faruq, D. Parker, and N. Hawes, “Probabilistic planning with formal performance guarantees for mobile service robots,” *The International Journal of Robotics Research*, vol. 38, no. 9, 2019.

[19] “Restriction of exposure - hazard control - 6.1.1 inverse square law,” University of St Andrews, accessed 2020-05-10. Available: https://www.st-andrews.ac.uk/staff/policy/healthandsafety/radiation/6restrictionofexposure-hazardcontrol/

[20] J. Duchi, “Derivations for linear algebra and optimization,” *Berkeley, California*, vol. 3, pp. 2325–5870, 2007.

[21] J. M. Kirkpatrick and B. M. Young, “Poisson statistical methods for the analysis of low-count gamma spectra,” *IEEE Transactions on Nuclear Science*, vol. 56, no. 3, pp. 1278–1282, 2009.

[22] S. Chakraborty, “Generating discrete analogues of continuous probability distributions-a survey of methods and constructions,” *Journal of Statistical Distributions and Application*, vol. 2, 08 2015.

[23] Z. Drezner and D. Zerom, “A simple and effective discretization of a continuous random variable,” *Communications in Statistics - Simulation and Computation*, vol. 45, no. 10, pp. 3798–3810, 2016. Available: https://doi.org/10.1080/03610918.2015.1071389

[24] B. Lacerda, D. Parker, and N. Hawes, “Optimal policy generation for partially satisfiable co-safe LTL specifications,” in *Proc. 24th International Joint Conference on Artificial Intelligence (IJCAI’15)*. IJCAI/AAAI, 2015, pp. 1587–1593.

[25] GPy, “GPy: A Gaussian process framework in Python,” http://github.com/SheffieldML/GPy, since 2012.

[26] A. G. d. G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman, “GPflow: A Gaussian process library using TensorFlow,” *JMLR*, vol. 18, no. 40, 2017.

[27] M. Kwiatkowska, G. Norman, and D. Parker, “PRISM 4.0: Verification of probabilistic real-time systems,” in *Proc. 23rd International Conference on Computer Aided Verification (CAV’11)*, ser. LNCS, vol. 6806. Springer, 2011, pp. 585–591.

[28] B. A. Khuwaileh and W. A. Metwally, “Gaussian process approach for dose mapping in radiation fields,” *Nuclear Engineering and Technology*, 2020. Available: https://www.sciencedirect.com/science/article/pii/S1738573319305935

[29] P. J. Diggle, J. A. Tawn, and R. A. Moyeed, “Model-based geostatistics,” *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 47, no. 3, pp. 299–350, 1998. Available: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9876.00113

[30] C. K. I. Williams and D. Barber, “Bayesian classification with Gaussian processes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1342–1351, 1998.

[31] L. Shang and A. B. Chan, “On approximate inference for generalized Gaussian process models,” 2013.

[32] Stanford Artificial Intelligence Laboratory et al., “Robotic Operating System.” Available: https://www.ros.org

[33] N. Hawes, C. Burbridge, F. Jovan, L. Kunze, B. Lacerda, L. Mudrova, J. Young, J. Wyatt, D. Hebesberger, T. Kortner, R. Ambrus, N. Bore, J. Folkesson, P. Jensfelt, L. Beyer, A. Hermans, B. Leibe, A. Aldoma, T. Faulhammer, M. Zillich, M. Vincze, E. Chinellato, M. Al-Omari, P. Duckworth, Y. Gatsoulis, D. C. Hogg, A. G. Cohn, C. Dondrup, J. Pulido Fentanes, T. Krajnik, J. M. Santos, T. Duckett, and M. Hanheide, “The strands project: Long-term autonomy in everyday environments,” *IEEE Robotics Automation Magazine*, vol. 24, no. 3, pp. 146–156, 2017.

Appendices

Department of Engineering Science

Supplementary Questions for 4th Year Project Students

Factor	Answer	Things to Consider	Record details here
Has the checklist covered all the problems that may arise from working with the VDU?	[✓ Yes] No		
Are you free from experiencing any fatigue, stress, discomfort or other symptoms which you attribute to working with the VDU or work environment?	[✓ Yes] No	Any aches, pains or sensory loss (tingling or pins and needles) in your neck, back shoulders or upper limbs. Do you experience restricted joint movement, impaired finger movements, grip or other disability, temporary or permanently	
Do you take adequate breaks when working at the VDU?	[✓ Yes] No	Periods of two minutes looking away from the screen taken every 20 minutes and longer periods every 2 hours Natural breaks for taking a drink and moving around the office answering the phone etc.	
How many hours per day do you spend working with this computer?	1-2 3-4 [✓ 5-7] 8 or more		
How many days per week do you spend working with this computer?	1-2 [✓ 3-5] 6-7		
Please describe your typical computer usage pattern	Working on a laptop with external mouse. Mixture of reading research papers and textbooks, and writing computer code.		

Student Declaration and Academic Approval	
Student Declaration: I have completed the DSE Workstation Checklist and the Supplementary Questions for my computer-related risk assessment for 4YP Project Number indicated below: 4YP Project Number: 11841 4YP Student's Name *(please print)* MATTHEW BUDD 4YP Student's Signature: [illegible]	Academic Approval I confirm my approval of this 4YP DSE Risk Assessment. Academic Supervisor's Name: *(please print)* Nick Hawes Academic Supervisor's Signature: NHawes