Spring 4-2017

# Regions of Inaccurate Modeling for Robot Anomaly Detection and Model Correction

Juan Pablo Mendoza
*Carnegie Mellon University*

# Regions of Inaccurate Modeling for Robot Anomaly Detection and Model Correction

Juan Pablo Mendoza

April 26, 2017
CMU-RI-TR-17-43

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

**Thesis Committee:**
Manuela Veloso, Co-Chair
Reid Simmons, Co-Chair
Jeff Schneider
Brian Williams, Massachusetts Institute of Technology

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

# Abstract

To make intelligent decisions, robots often use models of the stochastic effects of their actions on the world. Unfortunately, in complex environments, it is often infeasible to create models that are accurate in every plausible situation, which can lead to suboptimal performance. This thesis enables robots to reason about model inaccuracies to improve their performance. The thesis focuses on model inaccuracies that are subtle –i.e., they cannot be detected from a single observation– and context-dependent –i.e., they affect particular regions of the robot's state-action space. Furthermore, this work enables robots to react to model inaccuracies from sparse execution data.

Our approach consists of enabling robots to explicitly reason about parametric Regions of Inaccurate Modeling (RIMs) in their state-action space. We enable robots to *detect* these RIMs from sparse execution data, to *correct* their models given these detections, and to *plan accounting for uncertainty* with respect to these RIMs.

To detect and correct RIMs, we first develop algorithms that work effectively online in low-dimensional domains. An execution monitor compares outcome predictions made by a stochastic nominal model, to outcome observations gathered during execution. The results of these comparisons are then used to detect RIMs of state-action space in which outcome observations deviate statistically-significantly from the nominal model. Our detection algorithm is based on an explicit search for the parametric region of state-action space that maximizes an anomaly measure; once the maximum anomaly region is found, a statistical test determines whether the outcomes deviate significantly from the model. To correct detected RIMs, our algorithms apply corrections on top of the nominal model, only in the detected RIMs, treating them as newly-discovered behavioral modes of the domain.

To extend this approach to high-dimensional domains, we develop a search-based Feature Selection algorithm. Based on the assumption that RIMs are intrinsically low-dimensional, but embedded in a high-dimensional space, this best-first search starts from the zero-dimensional projection of all the execution data, and searches by adding the single most promising feature to the boundary of the search tree. Our low-dimensional algorithms can then be applied to the resulting low-dimensional space to find RIMs in the robot's planning model.

We also enable robots to make plans that account for their uncertainty about the accuracy of their models. To do this, we first enable robots to represent distributions over possible RIMs in their planning models. With this representation, robots can plan accounting for the probability that their models are inaccurate in particular points in state-action state. Using this approach, we enable robots to effectively trade off actions that are known to produce reward with those that refine their models, potentially leading to higher future reward.

We evaluate our approach on various complex robot domains. Our approach enables the CoBot mobile service robots to autonomously detect inaccuracies in their motion models, despite their high-dimensional state-action space: the CoBots detect that they are not moving correctly in particular areas of the building, and that their

wheels are starting to fail when making turns. Our approach enables the CMDragons soccer robots to improve their passing and shooting models online in the presence of opponents with unknown weaknesses and strengths. Finally, our approach enables a NASA spacecraft landing simulator to detect subtle anomalies, unknown to us beforehand, in their streams of high-dimensional sensor-output and actuator-input data.

# Acknowledgments

Working through my PhD has been a challenging, rewarding, and overwhelmingly joyful experience, due largely to the wonderful people around me.

I am very thankful to my advisors, Manuela Veloso and Reid Simmons, for the countless hours of discussion, agreements and disagreements, that lead me and my thesis to grow. In particular, I want to thank Manuela for never losing sight of the big picture, and always pushing me to remember that, to go beyond mathematical exercises, the work needs to be about real robots in the real world. And I want to thank Reid for being willing to question every detail and challenge every assumption of our algorithms. Many thanks to both of them for the thorough help and feedback with every algorithm, paper, and presentation, to ensure that it can endure the toughest scrutiny.

I am very grateful to my labmates for always creating an environment of positive collaboration and constructive criticism. In particular, I am extremely grateful to the CMDragons, with whom I spent many days and nights, growing together to understand how challenging it is for robots to work robustly all the time, and not just beyond lab demonstrations.

I want to thank my thesis committee members, Jeff Schneider and Brian Williams, for the highly meaningful discussions that helped me understand and convey the extents to which this thesis could apply to a variety of robot problems.

Finally, I am extremely grateful to my friends and my family, who have made my PhD years the happiest years of my life, so far.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Robots often use models of the effect of their actions on the world to make intelligent decisions throughout execution. Having accurate models enables robots to choose the right actions to perform their tasks effectively. Unfortunately, in many realistic environments, it is infeasible to have the perfect knowledge and computational resources required to create *globally accurate models* –i.e., models that accurately predict the effect of each action in every situation. Instead, these models may predict the true dynamics of the world relatively accurately in most circumstances, but fail to capture the dynamics of the world in some specific sets of similar situations.

This thesis addresses the problem of enabling robots to autonomously detect and react to such model inaccuracies, leading to significant improvement in execution performance. In essence, our approach consists of detecting these model inaccuracies, improving the models by applying corrections over the detected inaccuracies, and enabling the robots to make plans that take into account their knowledge and their uncertainty about possible inaccuracies in their models.

Section 1.1 introduces the specific problem on which this thesis focuses: enabling robots to act robustly in the presence of subtle and context-dependent model inaccuracies. Section 1.2 motivates the need to address this problem with three specific real robot examples, which we use throughout the thesis for evaluation of our approach. Section 1.3 describes this thesis' approach to solve the thesis problem: explicit reasoning about parametric Regions of Inaccurate Modeling (RIMs). Section 1.4 lists the specific technical contributions of this thesis. Finally, Section 1.5 gives an outline for the remainder of the thesis document.

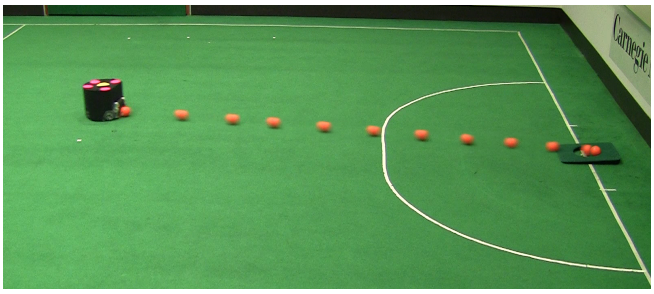
## 1.1 Thesis problem statement

This section describes the problem that this thesis addresses:

**Problem statement:** *To enable robots to act robustly, given sparse execution data, in the presence of subtle and context-dependent inaccuracies in their stochastic models.*

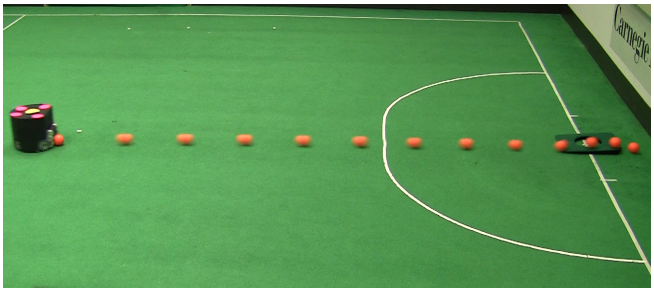
We proceed to describe the components of this problem statement using the illustrative example of Figure 1.1. In this illustrative domain, a robot is tasked with repeatedly putting a golf ball into the hole on the right edge of the field, with randomized starting locations for each shot. The robot has a single "shoot" action, so the state-action space of the robot is two-dimensional, and thus

easily visualizable. Throughout this thesis, we refer to the robot's state-action space, or a feature space derived from it, as the **context space** of the robot. Immediately after each shot, the robot observes precisely whether it succeeded or failed at scoring; throughout this thesis, we refer to the set of possible outcomes as the robot's **outcome space**. Thus, at each time step of execution, the robot receives a **contextual outcome** point consisting of a pair of a context –e.g., where the golfing robot shot from– and an outcome –e.g., whether the golfing robot succeeded.

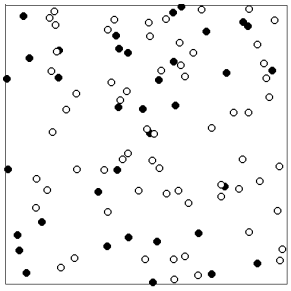


(a) Successful shot, nominal execution

(b) Nominal succes distribution



(c) Failed shot, nominal execution



(d) Nominal samples

Figure 1.1: Simple golf-putting task. Shots stochastically (a) succeed or (c) fail. (b) The robot builds or receives a nominal model of success probability over the field (lighter shows higher probability). (d) Simulated samples of success (white circles) and failure (black circles).

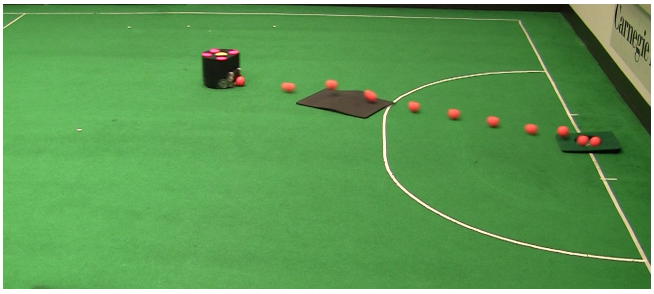
### 1.1.1 Stochastic models

This thesis focuses on domains in which the robots have a model of the expected behavior of the world, built from some combination of human design and training data. These models predict the distribution over outcomes, given a context point. For the example of Figure 1.1, the robot builds a model of its probability of scoring as a function of the location on the field from which it needs to shoot (Figure 1.1b). Depending on the domain, this model could be one of various predictive models: the transition function in a Markov Decision Process (MDP) [4], in which case the outcome space is the state space; a factor of the transition function in a factored MDP [23], in which case the outcome space is a subset of the state space; the reward function in a Contextual Multi-Armed Bandit (C-MAB) problem [52], in which case the outcome space is the reward space. Importantly, in real robot domains, these models are *stochastic* –i.e., they model a distribution over outcomes given a robot state and action.

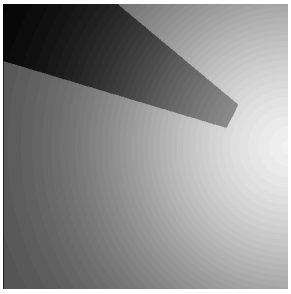
### 1.1.2 Subtle and context-dependent model inaccuracies

In complex robot domains, it is often infeasible to build globally accurate models at training time: the state-action space of the robot may be too large to explore completely at training time, or the deployment environment of the robot may need to be different from the environment used for training, or limited computational capabilities may require simple models. Due to this infeasibility, the robot's models may be partially inaccurate. We focus on domains in which these model inaccuracies are subtle and context-dependent, as described below.

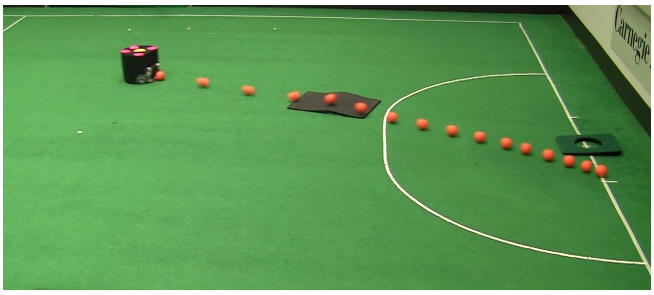
**Context-dependent model inaccuracies.** At deployment time, an unexpected and imperceptible bump on the field (Figures 1.2a and 1.2c) significantly reduces the robot's chances of successfully putting the ball whenever it shoots from behind the unseen bump (Figure 1.2b), creating a significant difference between its model's predictions and the observed execution from behind the bump (Figure 1.2d). Throughout this thesis, we use the term *context-dependent model inaccuracy* to refer to this type of inaccuracy, which affects the robot's performance in a particular region of its context space.



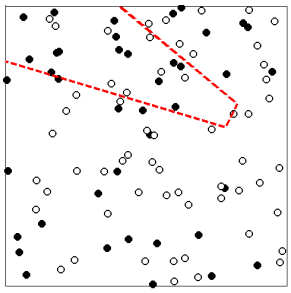
(a) Successful shot, anomalous execution



(b) Anomalous distribution



(c) Failed shot, anomalous execution



(d) Anomalous samples

Figure 1.2: (a,c) An imperceptible bump on the field causes a significant inaccuracy in the model of Figure 1.1b with respect to the true distribution of (b). (d) Synthetic samples behind the imperceptible obstacle show a significant deviation from the model of Figure 1.1b.

**Subtle model inaccuracies.** Even though the bump of Figure 1.2 affects the robot's performance significantly in a particular set of contexts, its effect is not large enough to be detected with certainty from any single outcome observation. Thus, the robot needs to analyze statistics of sets of

correlated contexts to determine that its execution does not match its nominal model. Throughout this thesis, we use the term *subtle model inaccuracies* to refer to this type of inaccuracy, whose effects are small enough that they need multiple execution data to be detected.

### 1.1.3 Sparse execution data

In many real robot domains, it is necessary for robots to react to model inaccuracies given only sparse data of their execution. There are several reasons for this need, which we explore in our different domains of interest:

**High dimensionality.** Robots usually have several components to their system, from high-level task planning to low-level motion and state estimation. Furthermore, they interact with the real world, which is highly complex. Thus, the full description of their state is usually high-dimensional, as exemplified in Section 1.2.1. This high-dimensionality implicitly makes it significantly harder to collect dense data, as the number of possible states scales exponentially with the dimension of the state space.

**Fast adaptation requirements.** In some robotics domains, fast adaptation in the presence of model inaccuracies is crucial. One such example is robot soccer, described in Section 1.2.2, in which only one match is played against a particular opponent. In robot soccer, adapting to different opponents' weaknesses and strengths over a single game, with sparse execution data, may mean the difference between a loss and a victory.

**Early detection requirements.** Early detection of anomalies is often critical for robust execution, as these anomalies may result in catastrophic robot failure if left unaddressed for longer periods of time, as exemplified in Section 1.2.3. Detecting these anomalies –in our case, model inaccuracies– early means detecting them from few, sparse data points.

### 1.1.4 Acting robustly in the presence of model inaccuracies

Depending on the target domain, there are several ways in which robots might act robustly in the presence of model inaccuracies. This thesis addresses the following approaches:

**Model inaccuracy detection.** In some domains, it suffices for the robot to detect that its execution is significantly different from nominal execution to increase its robustness. With detection, for example, the robot may stop execution and alert a human supervisor, who might then investigate the cause of the anomaly.

**Model inaccuracy characterization.** Aside from detecting a context-dependent model inaccuracy, the robot can provide a characterization of the inaccuracy by approximately determining the set of contexts it affects. This information may also help humans understand the root cause of the execution anomaly. For example, handing a human an approximation to the region shown in Figure 1.2d can help the human understand that there might be an unperceived obstacle in a particular location of the field.

**Model inaccuracy correction.** In some domains, such as those in which no human intervention is possible, detecting and characterizing a model inaccuracy does not suffice to increase

execution robustness. In such domains in which the robot needs to continue execution despite having detected an anomaly, the robot can apply a correction to its nominal model to account for the detected inaccuracy. Applying such corrections can lead the robot to choose different actions, thus improving its overall task performance.

**Planning accounting for model inaccuracy uncertainty.** Given the sparsity of execution data in our domains of interest, the robot may be unsure about whether a model inaccuracy exists, what its exact effect is on execution, and what is the exact set of contexts affected by it. The robot can create plans that take into account this uncertainty to effectively balance information-gathering actions to reduce that uncertainty with reward-seeking actions, with the goal of maximizing overall performance.

Throughout this thesis, we explore domains with different robustness requirements, and tackle the corresponding subproblems above accordingly.

## 1.2 Motivating domains

While the golf-putting robot example will provide an easily visualizable and controllable environment throughout the thesis, here we present the complex real robot domains that motivate our work. We describe the CoBot mobile service robot domain in Section 1.2.1, the CMDragons autonomous robot soccer domain in Section 1.2.2, and the NASA spacecraft landing simulator in Section 1.2.3. All these domains share the three criteria: stochastic nominal models, context-dependent model inaccuracies, and sparse execution data. However, they each present distinct challenges which we address.

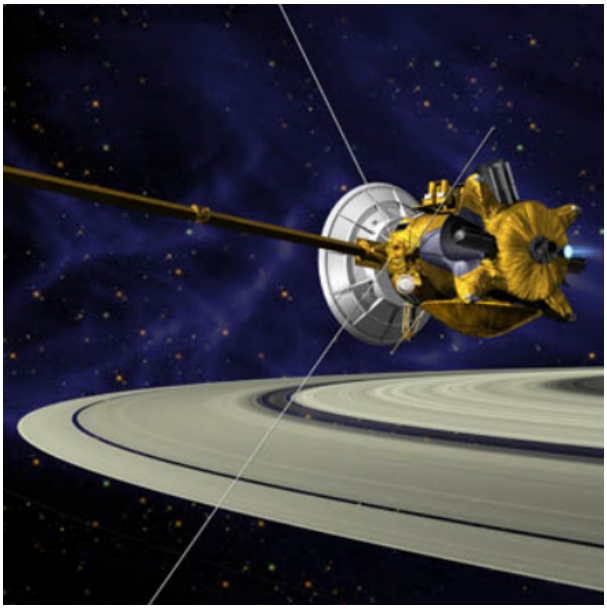
### 1.2.1 CoBot: Motion anomalies

Autonomous navigation is a key component of all mobile autonomous robots, and thus having an accurate motion model is essential for task performance. We explore the problem of context-dependent model inaccuracies in the motion of the CoBot autonomous service robots [95] of Figure 1.3a.

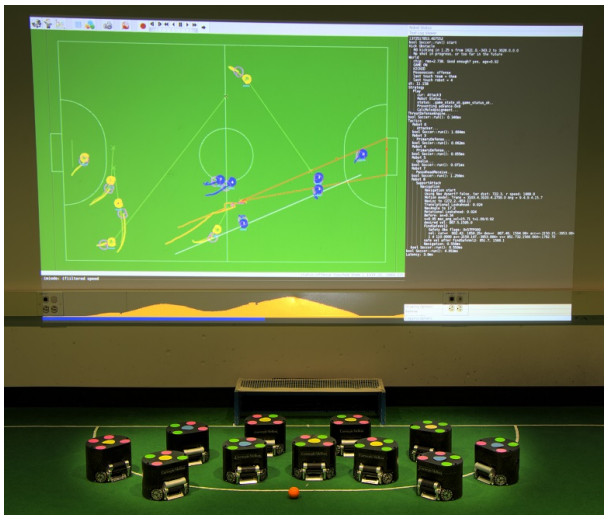
**Stochastic nominal model.** The CoBots autonomously perform tasks for the inhabitants of multiple buildings at Carnegie Mellon University (CMU). To accomplish high-level tasks, the CoBots navigate autonomously around the building, which entails moving and localizing properly [8]. For both of these tasks, the CoBots use a stochastic model that predicts the distribution over robot velocities at the next time step, given its velocity at the current time step and the motion commands given to its wheels.

**Subtle and context-dependent model inaccuracies.** While the CoBot's motion model is reasonably accurate in almost all circumstances, there are some situations in which it is not; due to the complexity of the real world, these situations are difficult to enumerate before deployment.

(a) CoBot service robots



(b) NASA Spacecraft

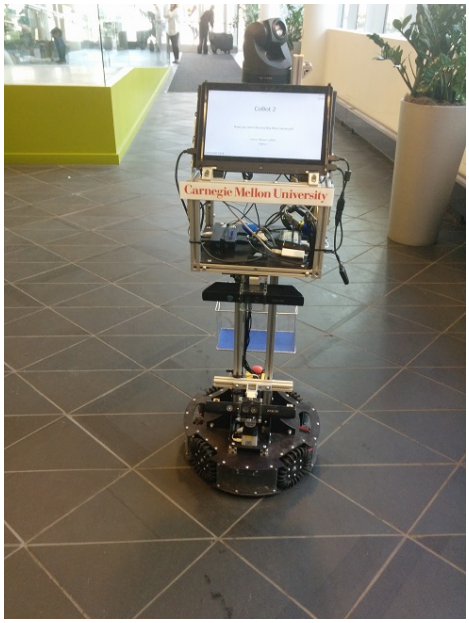


(c) CMDragons soccer robots

Figure 1.3: Robots that serve as motivation and application domains for our research. In different ways, these robots each need to react to subtle context-dependent model inaccuracies from sparse execution data.

Figure 1.4 shows two examples of such inaccuracies. Figure 1.4a shows an infrequent, but failure-inducing event in CoBot execution: there are certain obstacles in the CoBot's domain that are not perceptible to the robot's sensors. In the absence of sensors that can detect every obstacle perfectly, collisions are infrequent but eventually inevitable. Figure 1.4b shows a more subtle context-dependent model inaccuracy that happens when the CoBot drives over rough tile floor. Driving on this floor at normal speeds can lead the CoBot's hardware to get damaged. While the CoBots are not equipped to directly perceive the obstacles of Figure 1.4a or the rough floor of Figure 1.4b, we enable the CoBot to detect the inaccuracies they cause on the robot's motion model.


(a) Collision against an imperceptible table




(b) Rough tile floor that can cause hardware damage

Figure 1.4: Two examples of motion model inaccuracies in the CoBot's domain. This thesis enables the CoBot to detect these and other anomalies in its motion.

**Sparse execution data.** Even though the CoBots have gathered vast data from autonomously navigating around the building for more than 1000 km [5], their data remain sparse in their overall state-action space due to the *high dimensionality of the domain*: In addition to the robot's position and velocity, there are many other dimensions that are irrelevant to these particular model inaccuracies, such as the robot's orientation, its current task and schedule, the time of the day, and many others. However, the robot does not know in advance which of these dimensions are relevant.

**Acting robustly.** Once these situations are detected and their contexts characterized, the robot can (a) stop execution immediately and notify human deployers, and (b) learn to avoid entering these unsafe contexts –e.g., locations in the building where collisions happen more frequently, or

the location and speeds at which the rough floor causes model inaccuracies [1].

### 1.2.2 CMDragons: Planning for unknown opponents

Figure 1.3c shows the CMDragons team of autonomous soccer robots [62]. These robots participate in the RoboCup Small Size League (SSL), against other teams of soccer robots, and have won the SSL more than any other team, most recently in 2015. Each game of robot soccer lasts 20 minutes, and the later stages of the tournament are single elimination. Therefore, an extremely challenging problem in this domain is to enable robots to adapt to previously unknown opponents, with unknown weaknesses and strengths, over the course of a single game.

**Stochastic nominal models.** To adapt successfully, our team starts each game with several layers of stochastic models of themselves and of the opponents. These models range from high-level marking capabilities of the opponents, to low-level robot and ball motion models. These models are stochastic as there are many hidden variables in the opponent's internal state for which our models cannot accurately account, as well as usual sensor and actuator noise. These layers of initial models are crucial for our team to be able to play competitively, as even state-of-the-art techniques for learning to play soccer from scratch need much more data than is feasible within the span of a single game (e.g., reinforcement learning for keepaway [88]).

**Subtle and context-dependent model inaccuracies.** The single-elimination nature of RoboCup prevents our team from forming perfect models about our opponents ahead of time. Instead, our robots begin execution with our best estimate of their behavior – e.g., they intercept a moving ball the same way our team intercepts a moving ball. However, these models are inevitably inaccurate in some situations: different teams have different weaknesses and strengths, unknown to our robots ahead of the game. For example, some teams are significantly better at intercepting moving balls in some situations than other teams. Given the high stochasticity of the robot soccer domain, it is often infeasible to detect from a single outcome observation whether our models are inaccurate. For example, an opponent failing to intercept a ball which our model predicted would be intercepted may reflect a one-time glitch in that robot's execution, or it may reflect a deficiency in their interception algorithms. Our robots thus need multiple outcome observations to detect these inaccuracies.

**Sparse execution data.** The sparsity of data in this domain stems from its *fast adaptation requirements*, which makes it a particularly challenging domain for online adaptation. Within the span of a game, our robots have very few outcome observations for most of their actions: in a single game, there are usually fewer than a hundred shots, free kicks, and passes. Furthermore, the state of the game is very high dimensional, with over 80 physical dimensions alone, not counting the internal state of the robots or the game.

[1] This potential extension to our work is further discussed in Chapter 7.

**Acting robustly.** In the robot soccer domain, simply detecting an inaccuracy does not increase the team's robustness. Thus, we enable the team to apply corrections to their models online, and continue execution. For example, being able to detect and correct inaccuracies during the game may significantly affect our team's passing policy: passes that are very likely to succeed against one team may be very unlikely to succeed against another. Furthermore, we enable the soccer robots to explore actions that may initially appear sub-optimal with the intent of finding opponent weaknesses.

### 1.2.3 Spacecraft landing: Early detection of sensor anomalies

Figure 1.3b shows the spacecraft that the NASA simulator emulates. The spacecraft has various sensors and actuators that enable it to autonomously land on other planets. Failure in these sensors and actuators can result in catastrophic mission failure, but detecting component degradation or failure early can prevent it. Thus, we seek to enable the spacecraft to detect anomalies in its data stream as early as possible during execution.

**Stochastic nominal models.** The robot does not have an analytical model to predict future outcome observations given previous sensor data and chosen actions. However, it does have access to a large database of sensor and actuator streams from past successful nominal landings. Based on these data, we enable the robot to build a data-driven stochastic model to predict future sensor readings given its history of observations and input to its actuators.

**Subtle and context-dependent model inaccuracies.** Perhaps the largest challenge of this domain is that the context-dependent inaccuracies in it are unknown to us, the algorithms developers and evaluators. Work on this project was performed in conjunction with researchers at the NASA Jet Propulsion Laboratories (JPL); the researchers at JPL injected various context-dependent model inaccuracies into the sensor streams of the spacecraft, which the algorithms then had to find autonomously.

**Sparse execution data.** The sparsity of data in this domain comes from two sources: the requirement for early detection, and the high dimensionality of the domain. Early detection of sensor anomalies is crucial in this domain, since it can make the difference between a recoverable failure and a catastrophic failure; thus it is important to develop a method that can detect anomalies from as few data points as possible. Furthermore, the spacecraft has a 25-dimensional array of sensors providing data at each timestep, along with a 4-dimensional actuator input; a significant challenge is to find the low-dimensional context in which the anomaly manifests itself.

**Acting robustly.** The primary goal of our work on the spacecraft domain is to detect anomalies in sensor streams early during execution. Thus, acting robustly in this domain entails detecting and characterizing model inaccuracies, and reporting them to a human supervisor.

## 1.3 Approach: Regions of Inaccurate Modeling (RIMs)

We address the problem of robust execution in the presence of subtle and context-dependent model inaccuracies, described in Section 1.1, with the following approach, described in detail below:

**Thesis statement:** *Enabling robots to reason about parametric* **Regions of Inaccurate Modeling (RIMs)** *in their state-action space can improve execution robustness from sparse execution data in domains with subtle and context-dependent model inaccuracies.*

**Regions of Inaccurate Modeling (RIMs).** We focus on domains in which robots have stochastic models of nominal behavior, and which may have context-dependent model inaccuracies. Furthermore, we assume that these model inaccuracies affect particular connected regions of the robot's state-action space. Thus, we define Regions of Inaccurate Modeling (RIMs) as regions in this context space in which execution outcomes collectively significantly deviate from the distributions specified by the robot's models. For example, the golf-putting robot's outcome observations are each individually not highly unlikely given its nominal model. However, when aggregating the data received from the region behind the imperceptible bump (red dashed line in Figure 1.2d), the collection of outcomes is highly unlikely given the nominal distribution: their collective observed probability of success is significantly lower than predicted by the nominal model. Throughout this thesis, we focus on model inaccuracies in which execution data has a shift in the mean with respect to the expected distribution –e.g., the golfing robot's probability of success is lower than expected. However, extending our approach to account for other types of inaccuracies –e.g., those that affect the distribution's variance– is straightforward.

**Parametric RIMs** In particular, this thesis focuses on parametric RIMs–i.e., those that can be described by a finite vector of parameters. Focusing on parametric regions offers several advantages, given our domains of interest: parametric approaches tend to converge to an approximate solution from fewer observations than non-parametric approaches, which is necessary to meet our sparse data requirements; parametric regions have a concise vector representation, which the robots can use to conduct optimization-based searches over regions, and which humans can use to obtain a rough understanding of the spatial extent of RIMs. On the other hand, parametric RIMs are unable to express many regions that non-parametric approaches could. These advantages and disadvantages are further discussed throughout this thesis.

**Improving robustness using RIMs.** We explore three ways in which explicit reasoning about RIMs can lead to robust execution. First, we enable robots to autonomously **detect the presence of RIMs** in their state-action space from sparse execution data; for detection, the robot conducts a search in the parameter space of possible regions to find the region that maximizes an anomaly value, given the data contained in that region, and the robot's nominal model. At the moment of detection, the robot could stop execution and alert a human supervisor. Alternatively, we enable robots to **correct their models based on detected RIMs**, so they can continue execution with a more accurate model of the world. Finally, we enable robots to **make plans that account for uncertainty about RIMs** in their context space. That is, before the robot is sure about the existence

of RIMs, or about their precise shape, we enable the robot to make decisions that effectively trade off choosing actions that it knows lead to high reward and those that refine its knowledge about potential RIMs in its domain.

## 1.4 Thesis contributions

We seek to enable robots to improve their performance through reasoning about RIMs at execution time. This thesis presents the following contributions: (1) passive detection and correction of RIMs in the robot's models, (2) autonomous extraction of domain dimensions that are relevant to RIM-detection in high-dimensional domains, and (3) exploration of the domain online to refine models with the goal of improving overall performance.

### 1.4.1 Detection and correction of RIMs in low-dimensional domains

This contribution consists of theory and algorithms for online detection and correction of RIMs in low-dimensional domains. This problem is challenging due to stochasticity in robot domains, potential subtlety of the model inaccuracies, and sparsity of data.

To address this problem, we borrow ideas from research on disease outbreak detection [53]: our algorithms search for parametric regions of state-action space which are most likely to have come from a different distribution than the one specified by the model. This approach is analogous to finding regions of a map in which disease cases are significantly higher than expected. These approaches can deal with data sparsity because such regions may extend over potentially large regions of the domain with only few observations, as long as these few observations provide statistical support for the hypothesis of a model inaccuracy.

Originally, these regions of inaccuracy were found through exhaustive search of a discrete set of options [53]. On the other hand, we present optimization-based algorithms that scale more efficiently to higher dimensions than does exhaustive search; furthermore, the online and incremental nature of our optimization approach makes it appropriate for domains in which data arrive sequentially, such as robot domains.

This contribution is presented in detail in Chapter 3.

### 1.4.2 Feature selection for RIM detection

This contribution consists of an approach for reliable detection and correction of RIMs in high-dimensional domains. While the approach described in Chapter 3 has shown to significantly improve performance of robots in domains with up to 8 dimensions [65], its performance degrades with dimensionality too quickly to directly apply to high-dimensional domains. We address this problem because many complex robot domains have significantly higher dimensionality. For example, in robot soccer, the physical dimensionality of the world, not accounting for game state or internal robot state, is higher than 80 dimensions. Similarly, the CoBot robots have sensors that produce data streams with hundreds of dimensions, and they interact with the complexity of the real world.

To address this high-dimensionality problem, we start with the assumption that RIMs in the robot's model are intrinsically low-dimensional, but embedded in a high-dimensional space. This assumption holds in the vast majority of real-world RIMs our robots have encountered; for example, the robot's motion may be inaccurate when going through a certain region of the building at a particular set of speeds (3D RIM); on the other hand, it is highly unlikely that the CoBot's motion is inaccurate in a way that depends on its position, orientation, linear and angular velocities, time of the day, day of the week, and distance to the closest human (9D RIM).

Based on this low-dimensional RIM assumption, Chapter 4 presents a search-based method for finding the best low-dimensional subspace of the robot's space in which to look for these RIMs. Experimental results show that this approach vastly outperforms the direct application of RIM-detection algorithms of Chapter 3 in high-dimensional domains.

This contribution is presented in detail in Chapter 4.

### 1.4.3 Online learning under RIM-uncertainty

This contribution is to enable robots to effectively reason about their uncertainty about potential RIMs in their models. Reasoning about this uncertainty enables robots to effectively trade-off exploration and exploitation, with respect to RIMs, when choosing an action. As a concrete example, the CMDragons may have observed that the opponent goalie is much worse than expected at blocking shots towards the left side of the goal, but all of the shots so far have been taken from points close to the opponent's goal, since that is when the CMDragons choose to shoot instead of passing. Given this opponent weakness, the CMDragons may choose to attempt to shoot at the left side from far away, instead of passing, to maximize their scoring. More generally, because of data sparsity, even after model inaccuracies are detected, several RIM-hypotheses might be consistent with the robot's data. Reasoning about the distribution of such hypotheses enables the robot to improve performance through exploration.

This thesis develops the necessary theory and algorithms for the robots to autonomously make these decisions in the presence of RIMs. First, we introduce an approach for representing uncertainty about the spatial extent of potential RIMs in state-action space, using a set of parametric regions to represent a distribution over the true spatial extent of the RIM in question. Given this approximation to the distribution of possible spatial extents, we develop the theory needed to estimate the expected value of the robot's outcome observations given its state and chosen action, as well as the robot's uncertainty about this expected value. Using these expectation and uncertainty estimates, the robot applies the Upper Confidence Bound (UCB) algorithm [52] to effectively trade off exploration and exploitation.

Experimental data shows that this active exploration in the presence of uncertainty significantly outperforms passive detection and correction of RIMs in terms of performance improvement.

This contribution is presented in detail in Chapter 5.

## 1.5 Thesis outline

The rest of this document is organized as follows:

**Chapter 2:** This chapter provides a precise mathematical formulation of the scope of domains on which our approach is applicable. Furthermore, it provides a mathematical formulation for the problems of RIM-detection, model correction, and planning in the presence of RIMs. Finally, the chapter presents a short summary of the approach this thesis takes to solve each of these problems.

**Chapter 3:** This chapter presents the technical details of our solution to the problem of RIM-detection and correction in low-dimensional robot domains. The chapter begins with an overview of the general monitoring procedure that enables the robots to detect and correct RIMs in their models. Then, the chapter provides the necessary background required for understanding of the approach. The chapter proceeds with a technical description of an algorithm for detecting a single RIM, followed by an extension to detect multiple RIMs in the same robot domain. We then present an approach to improve robot models online by applying simple corrections to detected RIMs. Finally, the chapter presents empirical validation of the effectiveness of the presented approach on the golf-putting simulation, on the CoBots' motion models, and on the passing models of the CMDragons.

**Chapter 4:** This chapter presents the technical details of our solution to the problem of RIM-detection and correction in high-dimensional robot domains. The chapter begins with an overview of the approach, followed by the background required to understand the approach. Then, the chapter presents the technical description of our search-based feature selection method for finding the correct subspace of the high-dimensional domain to detect RIMs. Finally, we present empirical validation on the golf-putting scenario, on the CoBot robots' motion models, and on the NASA spacecraft simulation, showing that this approach vastly improves the detection power of RIM-detection.

**Chapter 5:** This chapter presents the technical details of our solution to the problem of enabling robots to plan in domains in which there is uncertainty about RIMs in their planning models. The chapter begins with an overview of the problem to solve and our approach. Then, the chapter presents the necessary background to understand our approach. Then, the chapter presents the technical description of our approach: first, we enable robots to represent their uncertainty about possible RIMs in their planning models; once robots can represent this uncertainty, they can obtain an estimate of the expected reward to be obtained by taking each action, as well as their uncertainty about that expectation; given these estimates, the robots use the well-established Upper Confidence Bound algorithm to trade off exploration and exploitation effectively.

**Chapter 6:** This chapter situates our work within the related research community. In particular, we compare our approach to existing approaches in the Execution Monitoring, Anomaly Detection, and Reinforcement Learning literature. The key distinguishing feature of our work is enabling robots to explicitly reason about parametric RIMs at execution time.

**Chapter 7:** This chapter concludes the thesis by providing a summary of the work presented here, a discussion of the implication of this work and possible directions for future related work.

# Chapter 2

# Problem Formulation and Approach

We seek to enable robots to act intelligently in the presence of Regions of Inaccurate Modeling (RIMs) in their models. This chapter formulates precisely the scope of domains on which our approach can operate, it formulates the problems that this thesis addresses, and it presents a high-level overview of the approach this thesis takes to address these problems. In short, our approach can be summarized by the following recurring steps:

1. A stochastic model of the world, which may be inaccurate in particular sets of similar contexts, provides information to the robot's planner.
2. The planner makes decisions to try to optimize expected performance; for this, the planner uses knowledge about RIMs in its planning models, and the robot's uncertainty about them.
3. The robot acts upon the world based on the decisions made by the planner.
4. The robot accurately observes the actual outcomes of its actions, and monitors how they differ from the expectations created by the planner.
5. If the monitor detects discrepancies between the robot's expectations and the observed outcomes, the robot reacts by either raising an anomaly alarm or applying a correction to its model and continuing execution.

Figure 2.1 shows the interaction among the different components involved in this process.

To formalize this approach, we start by precisely specifying the scope of domains to which our approach is applicable in Section 2.1. We then formalize the concept of context-dependent model inaccuracies in Section 2.2. Section 2.3 gives an overview of our approach to detecting these inaccuracies as RIMs. Finally, Section 2.4 gives an overview of how we enable robots to use these detected RIMs to improve their decision-making process.

## 2.1 Domain formulation

This section formalizes the scope of domains on which our approach can operate. In particular, we introduce a general concept of a *stochastic outcome model* that makes our approach applicable to a variety of models commonly used in robotics, such as Markov Decision Processes (MDPs), factored MDPs, and reward models in Contextual Multi-Armed Bandit problems (C-MABs). Further-

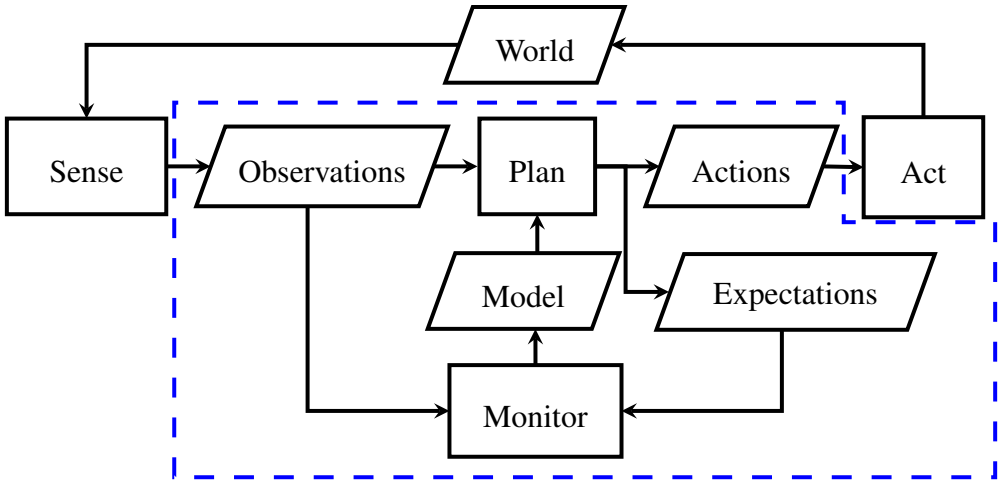



Figure 2.1: Structure of our robot architecture, including a Monitor module that modifies the robot's model. Our work focuses on the components enclosed in the blue dashed lines.

more, we illustrate some ways in which these models can be created in robot domains. Table 2.1 summarizes the most commonly used notation in this chapter and in the rest of the thesis.

The domains on which our approach can operate share these characteristics:

**Continuous states and actions.** In the domains of interest for this thesis, a robot $\rho$ needs to perform tasks in a continuous state space $\boldsymbol{S} \subseteq \mathbb{R}^{d_s}$, by choosing from a continuous space of actions $\boldsymbol{A} \subseteq \mathbb{R}^{d_a}$ to apply on the world. At each time step $t$, the robot is in a known state $\boldsymbol{s}_t$, and chooses apply action $\boldsymbol{a}_t$.

**Stochastic outcomes.** Applying action $\boldsymbol{a}_t$ in state $\boldsymbol{s}_t$ yields an observable *outcome* $\boldsymbol{z}_t \in \boldsymbol{Z}$. This outcome is stochastic, and follows an unknown distribution $P^*(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a})$, which depends only on the robot's state and the selected action.

**Stochastic outcome models.** The robot begins execution with a model $\boldsymbol{\theta}^0$ of the distribution of outcomes given a state and action:

$$\boldsymbol{\theta}^0(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) = P_{\boldsymbol{\theta}^0}(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a}) \tag{2.1}$$

Ideally, for every state, action and outcome, $P_{\boldsymbol{\theta}^0}(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a})$ closely approximates $P^*(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a})$. However, this thesis addresses domains in which, in some regions of state-action space, $P_{\boldsymbol{\theta}^0}(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a})$ significantly deviates from the true distribution $P^*(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a})$.

### 2.1.1 Application to common robotics models

The general definition of stochastic outcomes $\boldsymbol{z} \in \boldsymbol{Z}$ and their stochastic outcome model $\boldsymbol{\theta}^0$ enables our approach to be applied to various types of robotics domains. Table 2.2 provides a guide for using our approach to detect and correct model inaccuracies for problems modeled as MDPs, factored MDPs, and C-MABs, described in detail below.

| Notation | Meaning |
|---|---|
| $\rho$ | Robot |
| $\boldsymbol{S}$ | State space |
| $\boldsymbol{s} \in \boldsymbol{S}$ | State |
| $\boldsymbol{A}$ | Action space |
| $\boldsymbol{a} \in \boldsymbol{A}$ | Action |
| $\boldsymbol{\mathcal{X}}$ | State-action feature space |
| $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$ | State-action features |
| $\chi$ | Feature-extracting function |
| $\boldsymbol{Z}$ | Outcome observation space |
| $\boldsymbol{z} \in \boldsymbol{Z}$ | Outcome observation |
| $\boldsymbol{Z'} = \boldsymbol{\mathcal{X}} \times \boldsymbol{Z}$ | Contextual outcome space |
| $\boldsymbol{z'} = (\boldsymbol{x}, \boldsymbol{z}) \in \boldsymbol{Z'}$ | Contextual outcome |
| $\boldsymbol{\Theta}$ | Model space |
| $\boldsymbol{\theta}^0 \in \boldsymbol{\Theta}$ | Nominal behavior model |
| $\boldsymbol{\theta}^+ \in \boldsymbol{\Theta}$ | Corrected model |
| $P^*(\boldsymbol{z}\|\boldsymbol{x})$ | True (unknown) outcome distribution |
| $P_{\boldsymbol{\theta}}(\boldsymbol{z}\|\boldsymbol{x})$ | Outcome distribution according to model $\boldsymbol{\theta}$ |

Table 2.1: Table of commonly used notation for this thesis.

| Domain type | Predictive model name | Outcome $\boldsymbol{z}$ | Model $\boldsymbol{\theta}^0(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z})$ |
|---|---|---|---|
| MDP | Transition function | state $\boldsymbol{s'}$ | $P_{\boldsymbol{\theta}^0}(\boldsymbol{s'}\|\boldsymbol{s}, \boldsymbol{a})$ |
| Factored MDP | Transition function factor | state component $\boldsymbol{s'_i}$ | $P_{\boldsymbol{\theta}^0}(\boldsymbol{s'_i}\|\boldsymbol{s}, \boldsymbol{a})$ |
| C-MAB | Reward distribution | reward $r$ | $P_{\boldsymbol{\theta}^0}(r\|\boldsymbol{s}, \boldsymbol{a})$ |

Table 2.2: Guide to apply the work of this thesis to domains modeled as Markov Decision Processes (MDPs), factored MDPs, and Contextual Multi-Armed Bandit problems (C-MABs)

**MDPs.** In MDPs, the state of the world evolves according to a stochastic transition function $P^*(\boldsymbol{s'}|\boldsymbol{s}, \boldsymbol{a})$ that represents the probability of transitioning from state $\boldsymbol{s}$ to state $\boldsymbol{s'}$ by applying action $\boldsymbol{a}$. In model-based approaches to MDPs, the robot explicitly builds an estimate $P_{\boldsymbol{\theta}^0}(\boldsymbol{s'}|\boldsymbol{s}, \boldsymbol{a})$ of this transition function. To map these problems to our formulation, we let the outcome $\boldsymbol{z}$ of applying action $\boldsymbol{a}$ at state $\boldsymbol{s}$ be the resulting state $\boldsymbol{s'}$, and the stochastic outcome model is the transition function estimate.

**Factored MDPs.** In many complex domains, the transition function estimate can be broken down into various components: $P_{\boldsymbol{\theta}^0}(\boldsymbol{s'}|\boldsymbol{s}, \boldsymbol{a}) = \prod_i P_{\boldsymbol{\theta}^0}(\boldsymbol{s'_i}|\boldsymbol{s}, \boldsymbol{a})$, where the conjunction of all factors $\boldsymbol{s'_i}$ is the resulting state $\boldsymbol{s'}$. To map these problems to our formulation, we let the outcome $\boldsymbol{z}$ be an individual $\boldsymbol{s'_i}$, and the stochastic outcome model is the corresponding factor in the transition function estimate.

**C-MABs.** In C-MAB problems, at each timestep $t$, the robot is given a context $\boldsymbol{s}_t$ and must choose an action $\boldsymbol{a}_t$, which yields an observable reward $r_t$. This reward is drawn from an unknown reward distribution $P^*(r|\boldsymbol{s}, \boldsymbol{a})$, and the robot's goal is to maximize its overall reward $\sum_t r_t$. An approach to solve these problems is to maintain an estimate $P_{\boldsymbol{\theta}^0}(r|\boldsymbol{s}, \boldsymbol{a})$ of the reward distribution. To map these problems to our formulation, we let the outcome $\boldsymbol{z}$ be the observed reward $r$, and the stochastic outcome model be the reward distribution estimate.

### 2.1.2 Examples of nominal models and stochastic outcomes

Here, we describe some ways in which nominal models $\boldsymbol{\theta}^0$ and their corresponding stochastic outcomes $\boldsymbol{z}$ may be generated in robotics.

Nominal models may be generated from *sensor redundancy*: Autonomous robots often possess various sensors that, while usually designed to provide feedback about different aspects of the robot's environment, often present overlap or redundancy in the information they provide. A few examples of such redundancy include:

**Duplicated sensor redundancy** The easiest way to obtain redundant sensor data is to equip robots with multiple copies of a sensor (e.g., two IMUs, two GPS sensors). Since these sensors provide identical data, comparing observations between them is simple, and may amount to translation and rotation between the sensors. We note that duplicated sensors are more useful at detecting sensor failures than execution anomalies.

**Odometry redundancy** Robots often possess several sources of odometry data. For example, the CoBot robots have wheel encoders, RGB-D image sequences, and Inertial Measurment Units (IMUs), all of which can be used to obtain estimates of displacement over time in the robot's local frame of reference.

**Range sensor redundancy** Range to objects in the world can be extracted from various sensors, such as depth cameras, stereo cameras, laser rangefinders, and radars. It is often possible to transform these observations into comparable ones – for example, Figure 2.2 shows how a depth image can be projected down to its 2D footprint, yielding it comparable to laser scan information [7].

The robot's models about the effects of its actions can also be used as nominal models for execution. Most of the work in this thesis is based on this type of monitoring. Two of examples of this type of monitoring are:

**Motion model** The motion model of the robot describes how its actuators respond given a certain input. We have monitored the motion of the CoBot [63, 64] by comparing its expected velocity given a motion command to the velocity measured by its wheel encoders. Figure 2.3 shows an example of what such comparison looks like for the CoBot.

**Task performance** The expected performance of a robot, as indicated by its planning module, may be compared to actual performance. Examples of this include monitoring the time to perform tasks of the CoBot, or the pass success rate of the CMDragons attempting to play a game of keepaway [66].

These examples show that both low-level and high-level models of the robot's actions can be

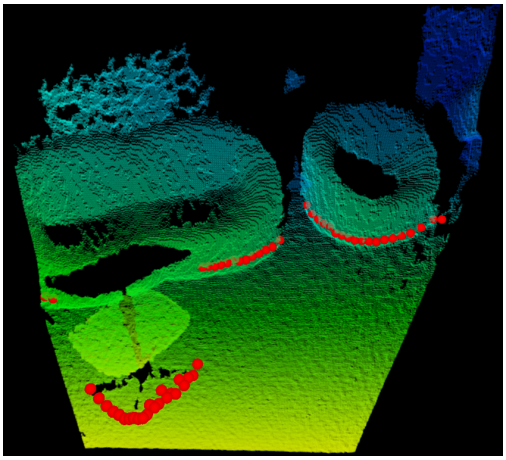
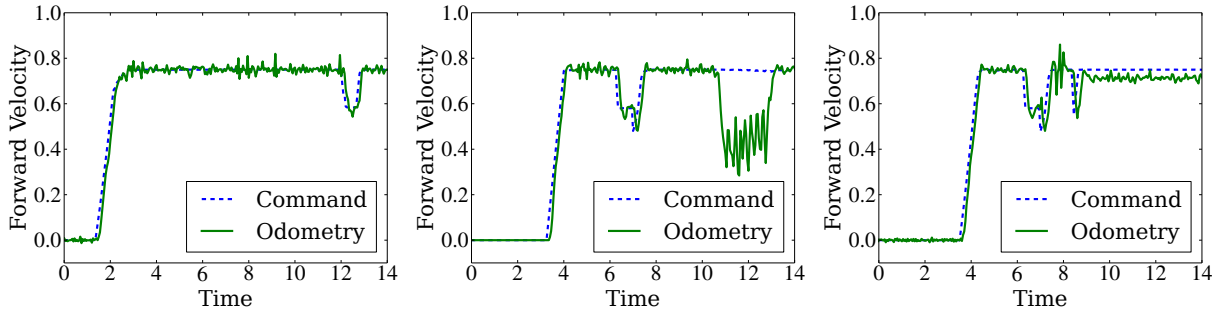


Figure 2.2: Projecting a depth image down to its 2D footprint yields it comparable to a laser scan. Thanks to Joydeep Biswas for the image based on his localization work [7].




(a) Nominal behavior (b) Collision (c) Subtle anomaly

Figure 2.3: Comparison between expected forward velocity deduced from the robot's motion model and commands, and the observed forward velocity as measured by wheel encoders.

monitored to produce measurable fit between models and execution.

We note that, in many cases, these comparisons between redundant sensors or between expected and observed action outcomes can be viewed as a process of generating *residuals*: a vector of values that, under nominal execution, are near zero. The problem of *generating robust residuals* that only deviate from 0 when execution is anomalous is beyond the scope of our thesis, but has been analyzed by previous research (see Section 6.1). Instead, we assume that these comparisons are noisy and focus on the problem of *evaluating them robustly*; these two approaches have been contrasted in previous survey work [39]. Section 2.3 describes our approach to evaluating these comparisons robustly.

## 2.2 Problem formulation

Ideally, the distribution $\boldsymbol{\theta}^0(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) = P_{\boldsymbol{\theta}^0}(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a})$ generated by the nominal model $\boldsymbol{\theta}^0$ would approximate the true distribution for all states and actions:

$$\text{Ideally: } \forall(\boldsymbol{s}, \boldsymbol{a}) \in (\boldsymbol{S} \times \boldsymbol{A}) \,.\boldsymbol{\theta}^0(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) \approx P^*(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a}) \tag{2.2}$$

However, it is often the case that a model is not a good predictor for every state and action. In particular, we are interested in domains in which there is some subset of the state-action space in which the nominal model $\boldsymbol{\theta}^0$ does not accurately describe execution. We focus on cases in which this inaccurate subset is made up of a finite set $\mathcal{R}^*_{SA}$ of connected regions of state-action space:

$$\text{Contextual inaccuracies: } \mathcal{R}^*_{SA} = \left\{ R^*_{SA} \subseteq (\boldsymbol{S} \times \boldsymbol{A}) | (\boldsymbol{s}, \boldsymbol{a}) \in R^*_{SA} \leftrightarrow \boldsymbol{\theta}^0(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) \not\approx P^*(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a}) \right\} \tag{2.3}$$

Many real-world domains have such connected regions of inaccuracy, as motivated in Section 1.2.

Furthermore, to maximize the generality of our approach, we allow these regions to be defined over *features* extracted from the state-action space by a function $\chi : \boldsymbol{S} \times \boldsymbol{A} \rightarrow \boldsymbol{\mathcal{X}} \subseteq \mathbb{R}^d$. This function might be handed to the robot, as in the work described in Chapter 3, or automatically extracted by the robot from a large set of possible features, as in the work described in Chapter 4. Under this more general definition, given $\boldsymbol{x} = \chi(\boldsymbol{s}, \boldsymbol{a})$, we define the finite set $\mathcal{R}^*$ as:

$$\text{Contextual inaccuracies (feature space): } \mathcal{R}^* = \left\{ R^* \subseteq \boldsymbol{\mathcal{X}} | \boldsymbol{x} \in R^* \leftrightarrow \boldsymbol{\theta}^0(\boldsymbol{x}, \boldsymbol{z}) \not\approx P^*(\boldsymbol{z}|\boldsymbol{x}) \right\}, \tag{2.4}$$

The existence, shape, or effect of these inaccurate regions $R^* \in \mathcal{R}^*$ are unknown to the robot. Our work consists on enabling robots to estimate these regions purely from sparse execution data.

## 2.3 Detection approach

This section gives an overview of our approach for detection of context-dependent inaccuracies in the robot's model.

The first approach that one may try is to compare individually each outcome observation received from the world to its expected distribution given by the nominal model. This simple approach is indeed capable of detecting some anomalous behavior in execution. For example, if a

robot's motion model indicated that the robot's next forward speed was expected to be distributed as $\mathcal{N}(1\frac{m}{s}, 0.1\frac{m^2}{s^2})$, but the robot's wheels are completely stuck and the next observation shows a forward speed of $0\frac{m}{s}$, the robot may immediately detect the anomaly since $P(v \leq 0|\mu = 1, \sigma^2 = 0.1)$ is vanishingly small. This thesis instead addresses the problem of subtle inaccuracies that cannot be detected using a single outcome observation.

Our approach is to enable the robot to explicitly search for Regions of Inaccurate Modeling (RIMs) of the context feature space $\boldsymbol{\mathcal{X}}$ in which execution data is *collectively* unlikely given the nominal model $\boldsymbol{\theta}^0$. The problem of detecting RIMs has been most widely studied in the context of disease outbreak detection (see Section 6.2.1). From this research, we borrow the general strategy for detecting such RIMs: First, we find the region of context space most likely to be anomalous –i.e., the region $R^+$ that maximizes the likelihood ratio:

$$\text{anom}(R, \boldsymbol{Z}, \boldsymbol{\theta}^0) = \frac{P\left(\boldsymbol{Z}|\text{execution in } R \text{ does not follow } \boldsymbol{\theta}^0\right)}{P\left(\boldsymbol{Z}|\text{execution in } R \text{ follows } \boldsymbol{\theta}^0\right)}. \tag{2.5}$$

Then, we conduct a statistical test to determine whether $R^+$ is, in fact, anomalous. This statistical test consists of testing whether $\text{anom}(R) > a_{\text{thresh}}$, for a derived threshold $a_{\text{thresh}}$ that corresponds to a desired detection power.

### 2.3.1 Detection in low-dimensional spaces

The search for $R^+$ cannot be conducted explicitly over all possibilities, since the set of all possible regions is infinite. In the disease outbreak community (e.g., [53]), this has most often been addressed by performing two approximations:

1. Limit the search space to a set of parametric regions of the domain.
2. Search over a discrete set of these parametric regions, often by discretizing the domain.

Here, we borrow the idea of restricting the search to a family of parametric regions. However, rather than explicitly searching over a discrete subset of these, we use optimization techniques to search for $R^+$. This approach scales better to domains with more than very few dimensions, and does not require discretization of a continuous domain. Furthermore, since observations arrive incrementally to the robot, solutions rarely change much from one time step to the next. Our optimization-based algorithms can take advantage of this by making slight modifications to the solution as new observations arrive, rather than restarting the search from scratch. One disadvantage of using optimization techniques is that, given the nonlinear nature of the problem, they cannot guarantee convergence to the globally optimal solution.

There are alternatives to finding $R^+$ other than searching over a space of parametric regions. For example, one could use regression trees to partition the domain into axis-aligned regions with similar behavior, or one could use non-parametric approaches, such as Gaussian Processes, to try to find RIMs. We have chosen to search over parametric regions because they provide a compact representation and fast convergence for non axis-aligned regions. Exploring alternate representations, while an interesting problem, is beyond the scope of this thesis.

Chapter 3 describes our work on detection of model inaccuracies in low-dimensional domains. We present an algorithm for detection of single and multiple RIMs during execution, and empirical

validation of these on the CoBot the CMDragons.

### 2.3.2 Detection in high-dimensional spaces

While the approach of Chapter 3 is effective in low- and mid-dimensional domains, its performance degrades quickly with the dimensionality of the domain. It is neither efficient nor effective to optimize RIMs in high-dimensional domains. To address this problem, we make a key observation: in the vast majority of context-dependent model inaccuracies, the intrinsic dimensionality of the inaccuracy is low; however, the robot does not know in advance which dimensions of its context space will be affected by model inaccuracies.

Assuming intrinsic low dimensionality in RIMs enables us to present a search-based approach over low-dimensional subspaces of the domain, to find the subspace that contains the maximum anomaly region. The search tree root node is the zero-dimensional subspace that contains no features, and the children of each node are the subspaces obtained by adding a single additional feature to the parent node. We propose various heuristics to enable the robot to efficiently search through this tree to find the subspace that contains the maximum anomaly region $R^+$.

Chapter 4 presents this best-first-search algorithm in detail, and provides empirical evidence of its effectiveness on real robot data.

## 2.4 Planning approach

Detected RIMs can be used by the robot's planner to improve its task performance. This thesis explores two aspects of this problem: model correction using RIM information, and planning that accounts for the robot's uncertainty regarding model inaccuracies.

### 2.4.1 Model correction

Robots can use information about detected RIMs to apply corrections to their models, which can in turn lead to substantial performance improvement. To correct the robot's model, we treat each detected RIM $R_i^+$ as a newly discovered *behavioral mode* of the system, with a different distribution of outcomes $\boldsymbol{\theta}^i$ than the nominal model $\boldsymbol{\theta}^0$.

The corrected model $\boldsymbol{\theta}^i$ in region $R_i^+$ is obtained as the maximum likelihood model out of a set of candidate models:

$$\boldsymbol{\theta}^i(R_i^+|\boldsymbol{Z}') \equiv \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left[ \prod_{\boldsymbol{x}_t \in R_i^+} P_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{x}_t) \right] \tag{2.6}$$

Then, the overall corrected model is given by a mixture of the models for each behavior:

$$\boldsymbol{\theta}^+ = \sum_i w_i \boldsymbol{\theta}^i. \tag{2.7}$$

The weights of this mixture model depend on the robot's estimated probability that the world will follow each of the behaviors, given the state in which it is located, and the action it chooses to take. Section 3.4 presents the technical details of this model correction approach.

### 2.4.2 Planning under RIM-uncertainty

In domains with sparse observations, multiple RIM-hypotheses might be consistent with the observations. Figure 2.4 illustrates a simple example of this in 2D: the observed data may be able to tell the robot with statistical significance that there exists a RIM, but not its exact shape and size.

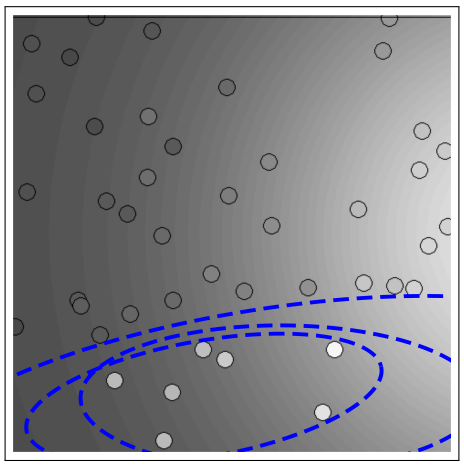


Figure 2.4: Observed outcomes during execution indicate the existence of a RIM. Blue ellipses show some of the infinite number of RIMs that are consistent with the observations.

We approach the problem of uncertainty about the extent of these RIMs by enabling the robot to represent each detected RIM as a distribution of possible parametric regions that are consistent with the RIM, rather than as a single parametric region. Once the robot represents each RIM as a set of possible parametric regions, it can reason more finely about whether a point in state-action space lies within the detected RIM: it can reason about the *probability* that a point lies within a detected RIM.

Representing this uncertainty enables our robots to effectively trade off exploration and exploitation. We develop an Upper Confidence Bound (UCB) algorithm, which has been applied in various domains in previous research as an effective algorithm for the Multi-Armed Bandit problem, to work within the context of RIM-detection and correction.

Chapter 5 presents the technical details of our approach to enable robots to reason about RIM-uncertainty, and an online learning algorithm for the Contextual Multi-Armed Bandit problem.

## 2.5 Chapter summary

This chapter presents a technical overview of the problem that this thesis addresses, and of the approach the thesis takes to solve this problem.

Section 2.1 provides a technical definition of the range of domains to which the work of this thesis applies: domains with continuous state and action spaces, in which the results of applying actions are stochastic, and in which the robot has a nominal model of the stochastic distribution

of these outcomes. The section shows that a wide variety of robotics domains fall under these restrictions.

Section 2.2 formally defines the problem this thesis solves: there exist exist particular regions of the robot's state and action space in which the distribution of stochastic action outcomes deviates significantly from the nominal stochastic model that the robot uses for planning.

Section 2.3 describes the general approach for detecting the existence and extent of such regions in which the robot's model is inaccurate. In low-dimensional domains, the robot uses an optimization-based approach to finding a parametric region of state-action space in which its model is inaccurate. In high-dimensional domains, the robot uses a search-based feature selection approach, in combination with the aforementioned low-dimensional approach, to find such regions effectively.

Section 2.4 describes how the robot can autonomously use this detection to improve its planning performance. First, once the robot has found a region of state-action space in which its model is inaccurate, the robot proceeds to apply a rough correction to its planning model in that particular region. Furthermore, the robot reasons about its uncertainty regarding possible model inaccuracies to improve its expected performance. Thus, the work on this thesis enables the robot to improve its task performance online by detecting, correcting, and reasoning about its uncertainty about model inaccuracies.

# Chapter 3

# Detection and Correction of RIMs in Low-Dimensional Spaces

This section describes our work on detecting and correcting Regions of Inaccurate Modeling (RIMs) in the robot's state-action space, during execution, in low-dimensional domains. The result is an execution monitoring framework that enables robots to detect and correct RIMs online.

Algorithm 1 shows a simplified description of this monitoring process, first described in our previous work [65]. At every time step $t$ of execution, this monitor receives as input the nominal model of execution $\boldsymbol{\theta}^0$, the state $\boldsymbol{s}_t$ visited by the robot, the action $\boldsymbol{a}_t$ taken by the robot, and the observed outcome $\boldsymbol{z}_t$ of that action. First, the algorithm extracts relevant features from state-action space (line 2); here, we assume the feature extraction function $\chi$ is given to the robot, but Chapter 4 describes our work to eliminate this assumption. Then, the contextual outcome observation $(\boldsymbol{x}_t, \boldsymbol{z}_t)$ is added to the set $\boldsymbol{Z}$ of all observations (line 3). The monitor proceeds to find RIMs fom $\boldsymbol{Z}$ and $\boldsymbol{\theta}^0$ (line 4), and correct the model accordingly (line 5).

**Algorithm 1** Execution monitor procedure run every time step $t$ of execution.
*Input:* model $\boldsymbol{\theta}^0$ of nominal execution, world state $\boldsymbol{s}_t$, chosen action $\boldsymbol{a}_t$ and observed outcome $\boldsymbol{z}_t$.
*Output:* corrected model $\boldsymbol{\theta}^+$.

1: **function** MONITOR($\boldsymbol{\theta}^0, \boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{z}_t$)
2: $\quad \boldsymbol{x}_t \leftarrow \chi(\boldsymbol{s}_t, \boldsymbol{a}_t)$ ▷ Extract relevant features
3: $\quad \boldsymbol{Z} \leftarrow \boldsymbol{Z} \cup (\boldsymbol{x}_t, \boldsymbol{z}_t)$ ▷ Update set of observations (initially, $\boldsymbol{Z} = \emptyset$)
4: $\quad \mathcal{R} \leftarrow \mathrm{FindRIMs}(\boldsymbol{Z}, \boldsymbol{\theta}^0)$ ▷ Find anomalies
5: $\quad \boldsymbol{\theta}^+ \leftarrow \mathrm{UpdateModel}(\boldsymbol{\theta}^0, \mathcal{R})$ ▷ Correct model
6: **end function**

The core of the algorithm thus lies in detecting and correcting model inaccuracies. We divide this problem into four components:

**Measure of inaccuracy:** Intuitively, we wish to detect and correct regions of state-action space in which observations do not match the robot's model. Because observations are stochastic, we define this measure statistically: we use a *likelihood ratio* measure, widely used in statistics and anomaly detection [14]. Section 3.1.1 describes this measure, and how we compute it.

**Search space of RIMs:** We would like to search over all possible regions of state space to find inaccurate ones. While some non-parametric methods can in fact represent arbitrarily-shaped boundaries, here we opt for parametric approaches, which usually converge sparser observations, at the cost of expressiveness power. Section 3.1.2 describes the parametric space we use in our completed work.

**RIM-search algorithm:** Given a search measure and space, we define the algorithm for searching over this space. Here, we deviate from previous related work, which conducts explicit searches, and take an optimization-based approach, which is better suited for robotics domains. Section 3.2 describes the FARO [64] algorithm for domains with up to one RIM, and Section 3.3 describes the DMAPS [65] extension for domains that may have multiple RIMs.

**RIM-correction:** Once RIMs are detected, our algorithms correct these inaccuracies to improve the robot's model. Section 3.4 describes how we address this in our work [65].

After describing each of these steps in detail, Section 3.5 describes how we evaluated these detection and correction algorithms in multiple simulated and real robot domains.

## 3.1 Background: Inaccuracy measure and search space

The problem of detecting RIMs has been explored in non-robotics contexts, such as disease outbreak detection. From this previous work [53], we borrow the measure used to decide whether a RIM is truly inaccurate, and the idea of using a particular family of parametric regions as the search space. However, here we derive the specifics to fit our problem domains.

### 3.1.1 Inaccuracy measure $\mathrm{anom}(R, \boldsymbol{Z}, \boldsymbol{\theta}^0)$

Following previous work, we define a RIM as a region $R$ such that the **likelihood of the data** observed in that region **given the robot's model** is much lower than the likelihood of it **given the best fitting model**. Thus, we use a **likelihood ratio** to define the inaccuracy measure[1] $\mathrm{anom}(R, \boldsymbol{Z}, \boldsymbol{\theta}^0)$ of a region $R$ given a set of observations $\boldsymbol{Z} = (\boldsymbol{x}_i, \boldsymbol{z}_i)$ and a nominal model $\boldsymbol{\theta}^0$:

$$\mathrm{anom}(R, \boldsymbol{Z}, \boldsymbol{\theta}^0) \equiv \frac{P(\boldsymbol{Z}|R \text{ follows } \hat{\boldsymbol{\theta}})}{P(\boldsymbol{Z}|R \text{ follows } \boldsymbol{\theta}^0)}, \tag{3.1}$$

where $\hat{\boldsymbol{\theta}}$ is the best fitting model from a set of possible alternate models $\hat{\boldsymbol{\Theta}}$. This statistical measure has various desirable detection properties, described in detail in previous work [53], including a variant of the uniformly most powerful property, and its independence from data that lies outside of $R$.

This inaccuracy measure depends only on the observations $(\boldsymbol{x}_i, \boldsymbol{z}_i)$ contained in $R$, and not explicitly on the shape of $R$ itself. This implicitly assumes that we have no prior information

[1] Because our work is strongly inspired by work on anomaly detection, we interchangeably use the terms *inaccuracy measure*, which reflects our specific purposes, and *anomaly measure*, which reflects the broader context for which this measure has been used.

about the likelihood of observing RIMs of particular shapes, which is a valid assumption in many robotics domains. Adding such prior information is possible, as shown in Section 3.3.

**Assumptions:** To calculate the anomaly value of a region $R$, we will make two assumptions.

- We assume a particular set of alternate models $\hat{\boldsymbol{\theta}}$: For all the states in a RIM, the mean $\boldsymbol{\mu}$ of the observations has been shifted by an unknown constant vector $\boldsymbol{\delta}$:

$$\mathbb{E}\left[\boldsymbol{z}|\boldsymbol{x},\hat{\boldsymbol{\theta}}\right] = \boldsymbol{\mu}(\boldsymbol{x}|\hat{\boldsymbol{\theta}}) = \begin{cases} \boldsymbol{\mu}(\boldsymbol{x}|\boldsymbol{\theta}^0) + \boldsymbol{\delta} & \text{if } \boldsymbol{x} \in R \\ \boldsymbol{\mu}(\boldsymbol{x}|\boldsymbol{\theta}^0) & \text{otherwise} \end{cases} \tag{3.2}$$

  While other work (e.g., [67]) assumes a multiplicative offset, an additive offset fits our domains best. In many monitoring applications, zero-mean observations (e.g., residuals) indicate normal execution; thus a multiplicative constant would not distinguish between normal and anomalous execution. This assumption results in a simple analytical form for Equation 3.1; in the future we may have to also consider inaccuracies that affect the variance of observations rather than the mean.

- We assume independence between observations $\boldsymbol{z}_i$ given the state of the robot $\boldsymbol{x}_i$. This assumption, which is valid given an expressive enough state description, allows us to decompose the probabilities of Equation 3.1 into individual probabilities $P(\boldsymbol{z}_i|\boldsymbol{x}_i, R, \boldsymbol{\theta})$.

With these assumptions, Equation 3.1 becomes:

$$\begin{aligned} \text{anom}(R, \boldsymbol{Z}, \boldsymbol{\theta}^0) &\equiv \frac{\max\limits_{\boldsymbol{\delta}} \prod\limits_{\boldsymbol{x}_i \in R} P(\boldsymbol{z}_i|\boldsymbol{\mu}(\boldsymbol{x}_i|\boldsymbol{\theta}^0) + \boldsymbol{\delta}) \prod\limits_{\boldsymbol{x}_i \notin R} P(\boldsymbol{z}_i|\boldsymbol{\mu}(\boldsymbol{x}_i|\boldsymbol{\theta}^0))}{\prod\limits_{\boldsymbol{x}_i} P(\boldsymbol{z}_i|\boldsymbol{\mu}(\boldsymbol{x}_i|\boldsymbol{\theta}^0))} \\ &= \frac{\max\limits_{\boldsymbol{\delta}} \prod\limits_{\boldsymbol{x}_i \in R} P(\boldsymbol{z}_i|\boldsymbol{\mu}(\boldsymbol{x}_i|\boldsymbol{\theta}^0) + \boldsymbol{\delta})}{\prod\limits_{\boldsymbol{x}_i \in R} P(\boldsymbol{z}_i|\boldsymbol{\mu}(\boldsymbol{x}_i|\boldsymbol{\theta}^0))} \end{aligned} \tag{3.3}$$

The above expression is a general measure of anomaly for arbitrarily shaped regions $R$ and for arbitrary distributions $P(\boldsymbol{z}|\boldsymbol{x})$. To make it more concrete, we now derive the expression for $\text{anom}(R)$ for the normal distribution $P(\boldsymbol{z}_i|\boldsymbol{x}_i, \boldsymbol{\theta}) \sim \mathcal{N}\left(\boldsymbol{\mu}(\boldsymbol{x}_i|\boldsymbol{\theta}), \boldsymbol{\Sigma}(\boldsymbol{x}_i|\boldsymbol{\theta})\right)$. We show the derivation for the normal distribution since it is one of the most commonly used. Analogous derivations for other distributions, and for multiplicative shifts (instead of our additive shift $\boldsymbol{\delta}$), can be found in related work [67]. For brevity, when discussing the nominal model $\boldsymbol{\theta}^0$, we use the abbreviations $\boldsymbol{\mu}_i \equiv \boldsymbol{\mu}(\boldsymbol{x}_i|\boldsymbol{\theta}^0)$, $\boldsymbol{\Sigma}_i \equiv \boldsymbol{\Sigma}(\boldsymbol{x}_i|\boldsymbol{\theta}^0)$.

For normal distributions, it is simpler to work with the logarithm of the likelihood ratio rather than with the likelihood ratio itself. Since logarithm is a monotone function, the region that maximizes one will maximize the other. Defining an auxiliary variable $\Delta\boldsymbol{z}_i \equiv \boldsymbol{z}_i - \boldsymbol{\mu}_i$, function

$F(R) \equiv \ln \text{anom}(R)$ is given by [2]

$$
\begin{aligned}
&= \max_{\boldsymbol{\delta}} \sum_{\boldsymbol{x}_i \in R} \left[ \ln(P(\boldsymbol{z}_i | \boldsymbol{\mu}_i + \boldsymbol{\delta}, \boldsymbol{\Sigma}_i) - \ln(P(\boldsymbol{z}_i | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \right] \qquad (3.4) \\
&= \max_{\boldsymbol{\delta}} \sum_{\boldsymbol{x}_i \in R} \frac{1}{2} \left[ \Delta \boldsymbol{z}_i^\top \boldsymbol{\Sigma}_i^{-1} \Delta \boldsymbol{z}_i - (\Delta \boldsymbol{z}_i - \boldsymbol{\delta})^\top \boldsymbol{\Sigma}_i^{-1} (\Delta \boldsymbol{z}_i - \boldsymbol{\delta}) \right] \\
&= \max_{\boldsymbol{\delta}} \sum_{\boldsymbol{x}_i \in R} \left[ \boldsymbol{\delta}^\top \boldsymbol{\Sigma}_i^{-1} \Delta \boldsymbol{z}_i - \frac{1}{2} \boldsymbol{\delta}^\top \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\delta} \right] \\
&= \max_{\boldsymbol{\delta}} \left[ \boldsymbol{\delta}^\top \sum_{\boldsymbol{x}_i \in R} \left( \boldsymbol{\Sigma}_i^{-1} \Delta \boldsymbol{z}_i \right) - \frac{1}{2} \boldsymbol{\delta}^\top \sum_{\boldsymbol{x}_i \in R} \left( \boldsymbol{\Sigma}_i^{-1} \right) \boldsymbol{\delta} \right] \qquad (3.5)
\end{aligned}
$$

To find $\boldsymbol{\delta}$ that maximizes this likelihood ratio we find the point where the derivative with respect to $\boldsymbol{\delta}$ is equal to 0:

$$
\begin{aligned}
\left( \sum_{\boldsymbol{x}_i \in R} \boldsymbol{\Sigma}_i^{-1} \Delta \boldsymbol{z}_i \right) - \left( \sum_{\boldsymbol{x}_i \in R} \boldsymbol{\Sigma}_i^{-1} \right) \boldsymbol{\delta}_{\max} &= 0 \\
\left( \sum_{\boldsymbol{x}_i \in R} \boldsymbol{\Sigma}_i^{-1} \right)^{-1} \left( \sum_{\boldsymbol{x}_i \in R} \boldsymbol{\Sigma}_i^{-1} \Delta \boldsymbol{z}_i \right) &= \boldsymbol{\delta}_{\max} \qquad (3.6)
\end{aligned}
$$

Substituting $\boldsymbol{\delta}_{\max}$ back into Equation 3.5 gives the final expression for the quantity to maximize:

$$
F(R) = \frac{1}{2} \left( \sum_{\boldsymbol{x}_i \in R} \boldsymbol{\Sigma}_i^{-1} \Delta \boldsymbol{z}_i \right)^\top \left( \sum_{\boldsymbol{x}_i \in R} \boldsymbol{\Sigma}_i^{-1} \right)^{-1} \left( \sum_{\boldsymbol{x}_i \in R} \boldsymbol{\Sigma}_i^{-1} \Delta \boldsymbol{z}_i \right) \qquad (3.7)
$$

This expression depends only on the nominal model $\boldsymbol{\theta}^0$ and sufficient statistics of the observed data. This is particularly useful when the state space is discretized and only sufficient statistics of each discrete state, instead of all the data points, need to be stored.

### 3.1.2 Search space of RIMs: Parametric regions

While we would like to search over the space of all possible subregions of state-action space to find RIMs, we choose to bound the search space to families of of parametric shapes. Previous work on spatial anomaly detection has used various parametric shapes (e.g., circles [53] or ellipses [54]). Previous work has argued that, while these regions only represent a small subset of all possible regions, their detection power extends significantly beyond the chosen shapes. Additionally,

[2] Unfortunately, capital letter $\Sigma$ is the standard symbol for both summations and covariance matrices. While we have kept this notation, we try to avoid confusion by always placing summation indices under $\Sigma$, while covariance matrices are indexed by a subscript to the right of $\Sigma$.

parametric shapes have the advantage of having an explicit concise parameter space over which to conduct a search.

Throughout our work, we do not assume a particular choice of parametric shapes, and our algorithms apply to any parametric shape (rectangles, ellipsoids, cylinders, etc.). The choice of shape depends on the domain at hand.

In our implementations, we have chosen to use ellipsoids as the search space for RIMs. Our motivation is that ellipsoids are a generic convex shape that support arbitrary rotations, translations, and scalings, while requiring only $O(d^2)$ parameters. In a $d$-dimensional state space, an ellipsoid can be parameterized by an $d$-vector $\boldsymbol{u}$ and a $d \times d$ positive definite matrix $\boldsymbol{A}$ as the set of points that satisfy:

$$(\boldsymbol{x} - \boldsymbol{u})^\top \boldsymbol{A}^{-1} (\boldsymbol{x} - \boldsymbol{u}) < 1 \tag{3.8}$$

Thus, the parameter vector $\boldsymbol{\psi}(\boldsymbol{u}, \boldsymbol{A})$ describing a particular ellipsoid is the linearized form of $\boldsymbol{u}$ and $\boldsymbol{A}$, consisting of $d + \frac{d(d+1)}{2} = \frac{1}{2}(d^2 + 3d)$ dimensions. The search space is then the space of such vectors $\boldsymbol{\psi} \in \boldsymbol{\Psi}$ such that the derived matrix $\boldsymbol{A}$ is positive definite.

## 3.2 Detection of a single RIM

In this section, we use the anomaly measure[3] $\mathrm{anom}(R)$ and the parametric search space defined in Section 3.1 to describe the Focused Anomalous Region Optimization (FARO) algorithm for detection of up to one RIM. While previous work in spatial anomaly detection has used similar measures and parametric shapes, we present a different algorithm, more suited for online monitoring of execution. As Section 6.2.1 discusses, previous search methods include exhaustive search over discretizations of the optimization space, and other methods that scale exponentially with the dimensionality of the data. Instead, we present an optimization-based approach.

Algorithm 2 describes this optimization process. It is an iterative optimization process that maintains a set $\mathcal{R}$ of most promising candidate RIMs throughout time, bounding the size of this set to a constant $|\mathcal{R}|_{\max}$. At every timestep of execution, the approach begins by seeding a new RIM candidate with a small region around the most recent observation (line 2). Then, the algorithm optimizes each candidate RIM using a nonlinear optimization algorithm (line 5). For this thesis, we have used the Cross-Entropy Method (CEM) of optimization, but we could replace it with another optimization method $\mathrm{Optimize}$, that takes as input a region $R$ (or its parameter vector $\boldsymbol{\psi}(R)$) and a value function $\mathrm{anom}(\cdot, \boldsymbol{Z}, \boldsymbol{\theta}^0)$ to be optimized.

Once every candidate RIM has been optimized, the algorithm must decide whether any of the candidates are statistically anomalous, and return that candidate as a RIM. This decision is based on comparing $\mathrm{anom}(R)$ to a threshold value $a_{\mathrm{thresh}}(t)$, which depends only on the domain during nominal execution, and on the number $t$ of observations (line 6). This threshold value can be approximately mapped to a desired rate of false positives in detection through Monte Carlo simulation [53]: we run FARO in $N$ simulated nominal executions, and count the number of

[3]When clear from context, we will ommit $\boldsymbol{Z}$ or $\boldsymbol{\theta}^0$ from $\mathrm{anom}(R, \boldsymbol{Z}, \boldsymbol{\theta}^0)$.

**Algorithm 2** FARO algorithm for detection of a single RIM.
*Input:* List of contextual observations $\boldsymbol{Z}$, nominal model $\boldsymbol{\theta}^0$, and maximum set size $|\mathcal{R}|_{\max}$.
*Output:* A RIM $R^+$, or $\emptyset$ if no RIM is detected.

**function** FARO( $\boldsymbol{Z} = [(\boldsymbol{x}_i, \boldsymbol{z}_i)|i = 0, \ldots, t]$, $\boldsymbol{\theta}^0$, $|\mathcal{R}|_{\max}$)
  $\mathcal{R} \leftarrow \mathcal{R} \cup \text{ball}(\boldsymbol{x}_t)$ ▷ Seed around latest $\boldsymbol{x}$. Initially, $\mathcal{R} = \emptyset$.
  $R^+ \leftarrow \emptyset$
  **for** $R \in \mathcal{R}$ **do**
    $R \leftarrow \text{Optimize}(R, \text{anom}(\cdot, \boldsymbol{Z}, \boldsymbol{\theta}^0))$ ▷ Nonlinear optimization. We use CEM
    **if** $\text{anom}(R) \geq a_{\text{thresh}}(t)$ **then** ▷ Statistical test for anomaly
      $R^+ \leftarrow R$
    **end if**
  **end for**
  **if** $|\mathcal{R}| > |\mathcal{R}|_{\max}$ **then**
    Reduce $\mathcal{R}$ to its $|\mathcal{R}|_{\max}$ elements of highest anom
  **end if**
  **return** $R^+$
**end function**

simulations $n(a_{\text{thresh}}, t)$ in which $\text{anom}(R^+) > a_{\text{thresh}}$ after $t$ observations:

$$P\left(\text{anom}(R^+) > a_{\text{thresh}}(t)|\boldsymbol{\theta}^0\right) \approx \frac{n(a_{\text{thresh}}, t)}{N}. \tag{3.9}$$

Figure 3.1 illustrates two examples of FARO detecting RIMs in the simulated golf-robot domain. Section 3.5 describes thorough evaluations of the detection capabilities of FARO.

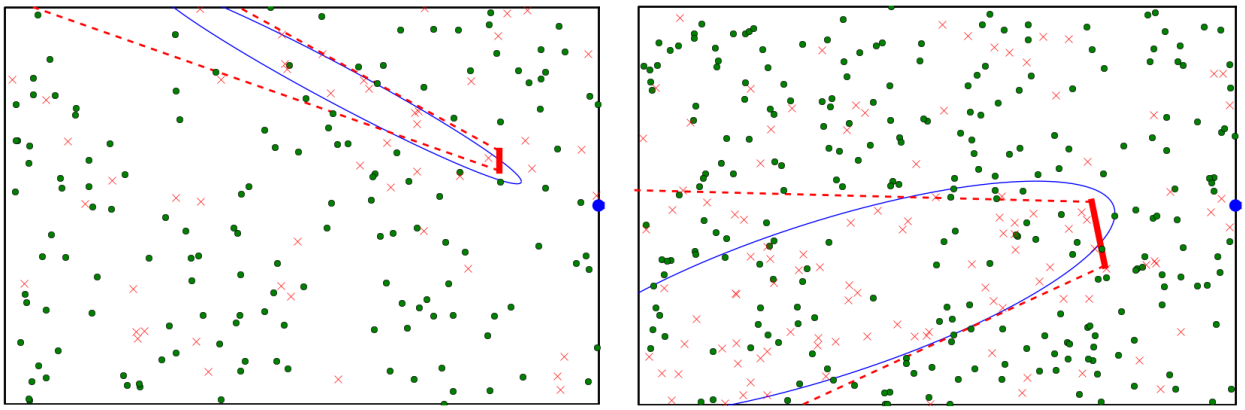
## 3.3 Detection of multiple RIMs

Section 3.2 describes our work on the FARO algorithm, created to detect RIMs in domains with up to one RIM. Here, we present the theory and algorithm needed to extend these ideas to domains that may have multiple RIMs. Section 3.3.1 derives the theoretical framework required for detection of multiple RIMs, while Section 3.3.2 puts this theory into the DMAPS algorithm for detection of multiple RIMs.

### 3.3.1 Multiple RIM detection theory

An intuitive (but ultimately insufficient) first idea on how to extend the formulation of Equation 3.1 to domains with multiple anomalies would be to simply maximize an analogous function $\text{Anom}'(\mathcal{R})$ over *sets $\mathcal{R}$ of regions*, rather than individual regions $R$:

$$\text{Anom}'(R, \boldsymbol{Z}, \boldsymbol{\theta}^0) \equiv \prod_{R \in \mathcal{R}} \frac{P(\boldsymbol{Z}|R \text{ follows } \hat{\boldsymbol{\theta}})}{P(\boldsymbol{Z}|R \text{ follows } \boldsymbol{\theta}^0)} \tag{3.10}$$



(a) $P(\boldsymbol{z}|\boldsymbol{s} \in \text{RIM}) = 0.2$. $P(\boldsymbol{z}|\boldsymbol{s} \notin \text{RIM}) = 0.8$ (b) $P(\boldsymbol{z}|\boldsymbol{s} \in \text{RIM}) = 0.5$. $P(\boldsymbol{z}|\boldsymbol{s} \notin \text{RIM}) = 0.8$

Figure 3.1: Synthetic data for a RIM (defined by red lines) created by an imperceptible bump (red solid line) in the golf-robot scenario. Green circles and red exes mark successful and failed shots, respectively. Blue ellipses show the RIMs found by FARO.

This formulation needs further work for two reasons: (1) overlap among regions needs to be properly addressed to avoid double counting of observations, and (2) we want our formulation to favor simpler hypotheses over more complex ones that try to explain the same observations. For example, using $\text{Anom}'(\mathcal{R})$ as the cost function, a hypothesis $\mathcal{R}$ with $n$ anomalous regions, each with one observation, would be considered as anomalous as a simpler hypothesis with one anomalous region with those $n$ observations.

To address the issue of double counting, we state the following natural assumption:

**Assumption 1.** *During execution, each observation in $\boldsymbol{Z}$ is produced by exactly one behavioral mode of the world. This mode could either be the nominal model, or one of the unmodeled models in the RIMs.*

In practice, what this assumption implies is that a single observation cannot contribute to the anomaly value $\text{anom}(R)$ of more than one region at a time during maximization.

The issue of favoring simpler hypotheses follows naturally from a proposed formulation of the optimization problem that accounts for prior probabilities over regions: instead of finding the set of regions that that maximizes $\text{Anom}'(\mathcal{R})$, we search for the set $R^+$ that maximizes the following function $\text{Anom}(\mathcal{R})$:

$$\text{Anom}\left(R, \boldsymbol{Z}, \boldsymbol{\theta}^0\right) \equiv \prod_{R \in \mathcal{R}} \frac{P\left(R \text{ follows } \hat{\boldsymbol{\theta}} | \boldsymbol{Z}\right)}{P\left(R \text{ follows } \boldsymbol{\theta}^0 | \boldsymbol{Z}\right)}, \tag{3.11}$$

which implicitly assumes that for any two regions $R_i \subseteq \boldsymbol{\mathcal{X}}$ and $R_j \subseteq \boldsymbol{\mathcal{X}}$, "$R_i$ is an anomalous region" is conditionally independent from "$R_j$ is an anomalous region", given the list of observations $\boldsymbol{Z}$. In domains in which $\boldsymbol{Z}$ is the only source of information about the presence of anomalous regions, this assumption holds. However, there may exist domains in which this assumption does

not hold: for example, a domain in which the robots knew in advance that there exist exactly $n$ anomalous regions. Addressing these domains is beyond the scope of this thesis.

The proposed formulation of Equation 3.11, leads to a more desirable optimization cost function, which naturally favors simpler hypotheses, as shown by Theorem 1 below:

**Theorem 1.** *The set $R^+$ of regions that maximizes $\mathrm{Anom}(\mathcal{R})$ is also the set that maximizes the simpler function*

$$F(\mathcal{R}) \equiv \left[\sum_{R\in\mathcal{R}} \log(\mathrm{anom}(R))\right] - \sum_{R\in\mathcal{R}} \lambda(R), \tag{3.12}$$

*with* $\lambda(R) = \log\left(\frac{P(R \text{ is normal})}{P(R \text{ is anomalous})}\right)$.

*Proof.* We can expand Equation 3.11 using Bayes' theorem:

$$\begin{aligned}
\mathrm{Anom}(\mathcal{R}) &= \prod_{R\in\mathcal{R}} \frac{P(R \text{ is anomalous}|\boldsymbol{Z})}{P(R \text{ is normal}|\boldsymbol{Z})} \\
&= \prod_{R\in\mathcal{R}} \frac{P(\boldsymbol{Z}|R \text{ is anomalous})P(R \text{ is anomalous})}{P(\boldsymbol{Z}|R \text{ is normal})P(R \text{ is normal})} \\
&= \left[\prod_{R\in\mathcal{R}} \mathrm{anom}\,(R)\right]\left[\prod_{R\in\mathcal{R}} e^{-\lambda(R)}\right].
\end{aligned} \tag{3.13}$$

Since $\log(\mathrm{Anom}(\mathcal{R}))$ is a monotonically increasing function of $\mathrm{Anom}(\mathcal{R})$, we maximize $\log(\mathrm{Anom}(\mathcal{R}))$ instead of $\mathrm{Anom}(\mathcal{R})$:

$$\begin{aligned}
\arg\max_{\mathcal{R}\subseteq 2^{\boldsymbol{\mathcal{X}}}} [\mathrm{Anom}(\mathcal{R})] &= \arg\max_{\mathcal{R}\subseteq 2^{\boldsymbol{\mathcal{X}}}} [\log(\mathrm{Anom}(\mathcal{R}))] \\
&= \arg\max_{\mathcal{R}\subseteq 2^{\boldsymbol{\mathcal{X}}}} \left[\left[\sum_{R\in\mathcal{R}} \log(\mathrm{anom}(R))\right] - \sum_{R\in\mathcal{R}} \lambda(R)\right] \\
&= \arg\max_{\mathcal{R}\subseteq 2^{\boldsymbol{\mathcal{X}}}} [F(\mathcal{R})]
\end{aligned} \tag{3.14}$$

□

**Corollary 1.** *In a domain in which the prior probability of region $R$ being anomalous is uniform for all $R \subseteq \boldsymbol{\mathcal{X}}$, $F(\mathcal{R})$ reduces to*

$$F(\mathcal{R}) = \left[\sum_{R\in\mathcal{R}} \log(\mathrm{anom}(R))\right] - \lambda|\mathcal{R}|, \tag{3.15}$$

*where $\lambda$ is a constant.*

For our completed work, we assume $\lambda$ to be a constant for all regions, although incorporating a non-uniform informative prior is a subject of interest for future work.

We note that, in domains in which RIMs are less probable than nominal behavior –i.e., $\lambda(R) > 0$– the cost function $F(\mathcal{R})$ naturally favors simpler hypotheses with fewer anomalous regions.

### 3.3.2 Algorithm for detection of multiple RIMs

Optimizing function $F(\mathcal{R})$ is a hard, non-convex problem. Even the simpler problem of globally optimizing for a single RIM is not tractable [64], which is why FARO uses an iterative optimization approach. Here, we also use an iterative optimization approach for each candidate RIM. Furthermore, our algorithm takes a greedy approach to optimize for multiple RIMs: candidate RIMs are optimized in sequence, in non-ascending order of their value $\mathrm{anom}(R)$.

Algorithm 3 describes the sequential optimization process of DMAPS. First, the algorithm adds a small candidate region around the most recent observation to $\mathcal{R}$ in line 2, similarly to the FARO algorithm. This set of regions is then sorted in non-increasing order of value $\mathrm{anom}(R)$ for sequential optimization. At this point, in line 7, the sequential optimization over candidate regions begins.

**Algorithm 3** DMAPS algorithm for detection of multiple RIMs.
*Input:* List of contextual observations $\boldsymbol{Z}$, and nominal model $\boldsymbol{\theta}^0$.
*Output:* A set $\mathcal{R}$ of potential RIMs, along with their corresponding anomaly values.

1: **function** DMAPS( $\boldsymbol{Z} = [(\boldsymbol{x}_i, \boldsymbol{z}_i)|i = 0, \ldots, t]$, $\boldsymbol{\theta}^0$)
2: $\quad\mathcal{R} \leftarrow \mathcal{R} \cup \mathrm{ball}(\boldsymbol{x}_t)$ ▷ Seed around latest $\boldsymbol{x}$. Initially, $\mathcal{R} = \emptyset$.
3: $\quad\mathbb{R} \leftarrow \mathrm{Sort}(\mathcal{R})$ ▷ Descending order of $\mathrm{anom}(R)$
4:
5: $\quad\mathcal{R} \leftarrow \emptyset$ ▷ Reset $\mathcal{R}$ and $F(\mathcal{R})$
6: $\quad F \leftarrow 0$
7: $\quad$**for** $R \in \mathbb{R}$ **do**
8: $\quad\quad R \leftarrow \mathrm{Optimize}(R, \mathrm{anom}(\cdot, \boldsymbol{Z}, \boldsymbol{\theta}^0))$ ▷ Nonlinear optimization. We use CEM
9: $\quad\quad$**if** $\log(\mathrm{anom}(R, \boldsymbol{Z})) \geq \lambda(R)$ **then**
10: $\quad\quad\quad\mathcal{R} \leftarrow \mathcal{R} \cup R$
11: $\quad\quad\quad F \leftarrow F + \log(\mathrm{anom}(R, \boldsymbol{Z})) - \lambda(R)$
12: $\quad\quad$**end if**
13: $\quad$**end for**
14: $\quad$**return** $\mathcal{R}$
15: **end function**

We note that this algorithm bears resemblance to the FARO algorithm. However, the results derived in Section 3.3.1 enable the algorithm to keep candidates only if they add to the total value of the set $R$.

For the optimization of each region $R$ in line 8, we use the Cross Entropy Method (CEM) for randomized optimization [81]. After optimizing a region from $R$ into $R'$, the decision of whether to add this region to the final output set is made in line 9. A region $R'$ is only added if it adds value to the optimization cost function $F$.

The end result of the DMAPS algorithm is a set of regions $R^+$ found to be most likely to be the set of unmodeled regions of the domain. Note that this set is not a global optimum of cost function $F(\mathcal{R})$ for two reasons: First, the CEM optimization of each region $R \in \mathcal{R}$ finds a locally optimal solution, rather than a globally optimal one. Second, the greedy sequential optimization

over multiple anomalies is not a globally optimal assignment either, and counter-examples to its optimality can be found. In spite of this, the detection of DMAPS, combined with the correction presented in Section 3.4 has shown to significantly improve performance and prediction in complex robot domains, as shown in Section 3.5.4.

## 3.4 Model correction using detected RIMs

Section 3.3 describes the DMAPS algorithm to find a set $\mathcal{R}$ of RIMs. This section describes how robots can use this information to correct their planning models. This corresponds to line 5 in Algorithm 1.

We seek to compute a corrected model $\boldsymbol{\theta}^+$ based on the nominal model $\boldsymbol{\theta}^0$ and the set $\mathcal{R}$ of detected anomalous regions:

$$P(\boldsymbol{z}|\boldsymbol{\theta}^+) \equiv P(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}^0, \mathcal{R}) \tag{3.16}$$

We assume that, when the robot performs action $a$ in state $s$, the world behaves according to its nominal model $\boldsymbol{\theta}^0$ or one of the $|\mathcal{R}|$ behaviors $\boldsymbol{\theta}^i$ detected by DMAPS. We denote the set of plausible models as $\boldsymbol{\Theta}^{\mathcal{R}} = \{\boldsymbol{\theta}^0\} \cup \{\boldsymbol{\theta}^i | i \in 1, \ldots, |\mathcal{R}|\}$. By the law of total probability, we obtain:

$$P(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}^+) = \sum_{\boldsymbol{\theta}^i \in \boldsymbol{\Theta}^{\mathcal{R}}} \left[ P(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}^i, \mathcal{R}) P(\boldsymbol{\theta}^i|\boldsymbol{s}, \boldsymbol{a}, \mathcal{R}) \right]. \tag{3.17}$$

The distribution of observations is a mixture of the distributions produced by the different plausible models. We get a better idea of the parameters involved in Equation 3.17 by defining $\alpha_i(\boldsymbol{s}, \boldsymbol{a}, \mathcal{R}) \equiv P(\boldsymbol{\theta}^i|\boldsymbol{s}, \boldsymbol{a}, \mathcal{R})$, and noting that $\boldsymbol{z}$ is independent of $\mathcal{R}$ given the right model $\boldsymbol{\theta}^i$:

$$P(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}^+) = \sum_{i=0}^{|\boldsymbol{\Theta}^{\mathcal{R}}|} \left[ \alpha_i P(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}^i) \right] \tag{3.18}$$

Section 3.4.1 discusses the problem of estimating the model $\boldsymbol{\theta}^i$, while Section 3.4.2 discusses the estimation of $\alpha_i$. Then, Section 3.4.3 describes how to bring these together into the final model correction equations.

### 3.4.1 Estimating observation distributions

$P(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}^i)$ describes the distribution of observations $\boldsymbol{z}$ given that the world is in a RIM defined by model $\boldsymbol{\theta}^i$. For nominal execution, this is simply the model-given predicted distribution $P(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}^0)$.

For the unmodeled behavior modes in each region $R_i \in \mathcal{R}$, we assume for this work that the form of the distribution is the same as in nominal execution, but the parameters are specific to each unmodeled behavior. Thus, in the absence of prior knowledge about the distribution of anomalies, the maximum likelihood estimate is used, as in Section 3.2:

$$\boldsymbol{\theta}^i \equiv \arg\max_{\hat{\boldsymbol{\theta}} \in \hat{\boldsymbol{\Theta}}} \left[ \prod_{(\boldsymbol{x}_i, \boldsymbol{z}_i, \boldsymbol{\theta}_i) \in \boldsymbol{Z}} \mathbb{1}(\boldsymbol{x}_i \in R) \frac{P(\boldsymbol{z}_i|\hat{\boldsymbol{\theta}}(\boldsymbol{x}_i))}{P(\boldsymbol{z}_i|\boldsymbol{\theta}_i)} \right] \tag{3.19}$$

In particular, for the examples of this work, we use Gaussian distributions and anomalies that shift the mean $\boldsymbol{\mu}$ of the distribution. In that case, the maximum likelihood distribution for model $\boldsymbol{\theta}^i$ is given by

$$P\left(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}^i\right) = \mathcal{N}\left(\boldsymbol{\mu}_0(\boldsymbol{s},\boldsymbol{a}) + \boldsymbol{\delta}(R_i), \boldsymbol{\Sigma}_0(\boldsymbol{s},\boldsymbol{a})\right). \tag{3.20}$$

Here, $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}_0$ are the parameters predicted by the nominal model $\boldsymbol{\theta}^0$, and $\boldsymbol{\delta}(R_i)$ is the maximum likelihood shift in mean of Equation 3.6.

### 3.4.2 Estimating active behavior distributions

Coefficients $\alpha_i = P(\boldsymbol{\theta}^i|\boldsymbol{s},\boldsymbol{a},\mathcal{R})$ describe the probability of the world being in each behavior model $\boldsymbol{\theta}^i$ (including nominal behavior) given that the robot takes action $\boldsymbol{a}$ in state $\boldsymbol{s}$. First we note that, for nominal behavior this probability is constrained to be

$$P\left(\boldsymbol{\theta}^0|\boldsymbol{s},\boldsymbol{a},\mathcal{R}\right) = 1 - \sum_{i=1}^{|\mathcal{R}|} P\left(\boldsymbol{\theta}^i|\boldsymbol{s},\boldsymbol{a},\mathcal{R}\right), \tag{3.21}$$

so we focus on computing the probability of other behaviors.

The activation probability of each unmodeled behavior $\boldsymbol{\theta}^i$ is given by:

$$\begin{aligned} P(\boldsymbol{\theta}^i|\boldsymbol{s},\boldsymbol{a},\mathcal{R}) &= P\left(\boldsymbol{\theta}^i|\chi(\boldsymbol{s},\boldsymbol{a}) \in R_i\right) P\left(\chi(\boldsymbol{s},\boldsymbol{a}) \in R_i\right) && (3.22)\\ &= P\left(R_i \text{ follows } \boldsymbol{\theta}^i\right) P\left(\chi(\boldsymbol{s},\boldsymbol{a}) \in R_i\right), && (3.23) \end{aligned}$$

where we have rewritten the first factor to make it clear the relationship to Equation 3.1 clear. The activation of behavior mode $\boldsymbol{\theta}^i$ thus depends both on whether feature state $\boldsymbol{x} = \chi(\boldsymbol{s},\boldsymbol{a}) \in \boldsymbol{\mathcal{X}}$ lies inside of region $R_i$, and on our confidence that $R_i$ is a RIM that behaves according to $\boldsymbol{\theta}^i$.

For our completed work, $P\left(\chi(\boldsymbol{s},\boldsymbol{a}) \in R_i\right)$ simply indicates whether a state-action point belongs to region $R_i$:

$$P\left(\chi(\boldsymbol{s},\boldsymbol{a}) \in R_i\right) = \mathbb{1}(\chi(\boldsymbol{s},\boldsymbol{a}) \in R_i). \tag{3.24}$$

In domains with noisy measurements of $\boldsymbol{s}$, or with anomalous regions with fuzzy boundaries, a softer definition of "belonging" to region $R_i$ could be more appropriate.

To calculate the confidence value $P\left(R_i \text{ follows } \boldsymbol{\theta}^i\right)$, we again employ Monte Carlo simulation, similarly to how we use it in Section 3.2: We note that $P\left(R_i \text{ follows } \boldsymbol{\theta}^i\right)$ is a monotonically increasing function of $\mathrm{anom}(R)$, as defined in Equation 3.1. There exists a function $\mathbb{P} : \mathbb{R} \to \mathbb{R}$ such that:

$$\mathbb{P}(\mathrm{anom}(R)) = P\left(R_i \text{ follows } \boldsymbol{\theta}^i\right). \tag{3.25}$$

This function can be estimated by sampling: If it is possible to have access to state-action samples distributed as those in $\boldsymbol{Z}$ and in the absence of anomalies, an empirical estimate $\hat{\mathbb{P}}$ can be created from the empirical estimate of $P(\mathrm{anom}(R) > \gamma | R \text{ follows } \boldsymbol{\theta}^0)$ [53]. Therefore, we estimate the desired confidence by:

$$P\left(b_R|\boldsymbol{x}(\boldsymbol{s},\boldsymbol{a}) \in R, \boldsymbol{Z}\right) = \hat{\mathbb{P}}(\mathrm{anom}(R)) \tag{3.26}$$

Coefficient $\alpha_R$ is thus computed as:

$$\alpha_R(\boldsymbol{s},\boldsymbol{a},\mathcal{R}) = \mathbb{1}\left(\boldsymbol{x}(\boldsymbol{s},\boldsymbol{a}) \in R\right) \hat{\mathbb{P}}(\mathrm{anom}(R)) \tag{3.27}$$

### 3.4.3 Applying model corrections

Defining $P(\boldsymbol{z}|\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}^i)$ and $\alpha_i$ for each plausible behavior fully defines Equation 3.18. Using normal distributions as described in Section 3.4.1 and $\alpha_i$ as defined in Section 3.4.2, we get the corrected model:

$$\begin{aligned} P(\boldsymbol{z}|\boldsymbol{\theta}^+) = \\ P(\boldsymbol{z}|\boldsymbol{\mu}_0,\boldsymbol{\Sigma}_0)\left(1-\sum_{R\in\mathcal{R}}\left[\mathbb{1}\left(\boldsymbol{x}(\boldsymbol{s},\boldsymbol{a})\in R\right)\hat{\mathbb{P}}(\text{anom}(R))\right]\right)+ \\ \sum_{R\in\mathcal{R}}\left[P\left(\boldsymbol{z}|\boldsymbol{\mu}_0+\boldsymbol{\delta}(R),\boldsymbol{\Sigma}_0\right)\mathbb{1}\left(\boldsymbol{x}(\boldsymbol{s},\boldsymbol{a})\in R\right)\hat{\mathbb{P}}(\text{anom}(R))\right] \end{aligned} \tag{3.28}$$

Given that $\boldsymbol{x}(\boldsymbol{s},\boldsymbol{a})$ can only be part of one anomalous region by Assumption 1, we get the final expression for the distribution of observations:

$$P(\boldsymbol{z}|\boldsymbol{\theta}^+) = \begin{cases} P(\boldsymbol{z}|\boldsymbol{\mu}_0,\boldsymbol{\Sigma}_0)(1-\hat{\mathbb{P}}(\text{anom}(R)))+ & \text{if } \exists R\in\mathcal{R} \text{ s.t.} \\ P(\boldsymbol{z}|\boldsymbol{\mu}_0+\boldsymbol{\delta}(R),\boldsymbol{\Sigma}_0)\hat{\mathbb{P}}(\text{anom}(R)) & \boldsymbol{x}(\boldsymbol{s},\boldsymbol{a})\in R \\ \\ P(\boldsymbol{z}|\boldsymbol{\mu}_0,\boldsymbol{\Sigma}_0) & \text{otherwise} \end{cases} \tag{3.29}$$

From this expression, the planner can estimate different statistics of the predicted distribution of $\boldsymbol{z}$. For example, the expected value of $\boldsymbol{z}$ can be easily derived as

$$\mathbb{E}[\boldsymbol{z}|\boldsymbol{\theta}^+] = \begin{cases} \boldsymbol{\mu}_0+\boldsymbol{\delta}(R)\hat{\mathbb{P}}(\text{anom}(R)) & \text{if } \exists R\in\mathcal{R} \text{ s.t. } \boldsymbol{x}(\boldsymbol{s},\boldsymbol{a})\in R \\ \\ \boldsymbol{\mu}_0 & \text{otherwise} \end{cases} \tag{3.30}$$

## 3.5 Empirical evaluation

Sections 3.1 through 3.4 describe our theory and algorithm contributions to online detection and correction of RIMs. This section describes how we evaluated each of these contributions.

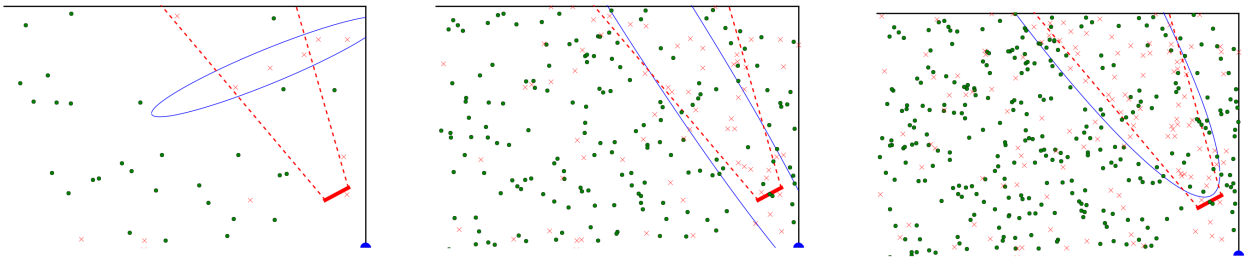
First, we present a summary of our evaluations of the FARO algorithm for online detection of domains which may have a single RIM; our previous work [64] presents a more detailed description of these evaluations. We evaluated FARO in two domains: synthetic data from the golf-robot domain (Section 3.5.1), and real-world data from the CoBot service robots [95] (Section 3.5.2). The objectives of experimenting in these two domains are that the first provides a simple and easily visualizable domain, while the second provides a real-world application with a higher dimensional state space; the two combined support the generality of the algorithm.

Then, we describe evaluations of the DMAPS algorithm, an the performance improvement achieved by applying model corrections; these evaluations are more thoroughly described in previous work [65]. We describe the evaluation domain of soccer Interception-Keepaway in Section 3.5.3, and then describe our experiments and results in Section 3.5.4.

### 3.5.1 Golf domain single RIM detection

We first show experiments and results using synthetic data from the golf-robot domain described in Section 1. Figure 3.2 shows this domain, in which the robot's task is to repeatedly shoot a ball into a hole on the far right side of the field. Shot observations arrive sequentially, and the locations of shots are uniformly distributed across the field.

The domain has one RIM created by an imperceptible bump on the field, which causes the robot to miss its target more often than expected when shooting from behind the bump.



(a) $\ln \text{anom}(R) = 10.0$, 7 observations

(b) $\ln \text{anom}(R) = 20.6$, 43 observations

(c) $\ln \text{anom}(R) = 40.8$, 84 observations

Figure 3.2: Most abnormal ellipse found as more observations arrive. The sub-figures show the state of the algorithm when three anomaly thresholds are reached. Parameter values are $p = 0.8$, $q = 0.5$.

The experimental setup is as follows: the field measures $6 \times 4$ meters. The parameters in this world are the shape and location of the unperceived obstruction $\text{obs}$, the probability $p$ of success during normal execution, and the probability $q$ of success from locations that are blocked by $\text{obs}$ (abnormal execution), both of which are constant throughout their regions. For each test of each of the experiments below, $\text{obs}$ was sampled uniformly from the set of line obstructions where the distance between the target and the center of $\text{obs}$ is between 0.1 and 1.5 meters; the center of $\text{obs}$ is in the field, and the angle subtended by the endpoints of $\text{obs}$ and the target is between $\frac{\pi}{16}$ and $\frac{\pi}{4}$ radians. This randomization created test RIMs of various shapes, sizes, and orientations.

The goal of the first experiment was to determine the effect of the number of available abnormal data points on the effectiveness of the detection. To do this, we held both $p$ and $q$ constant and kept track of the precision and recall rates of the algorithm as the number of abnormal data points increased. In the context of this work, given a ground truth anomalous region $S$, precision and recall of the estimated region $R$ are defined as

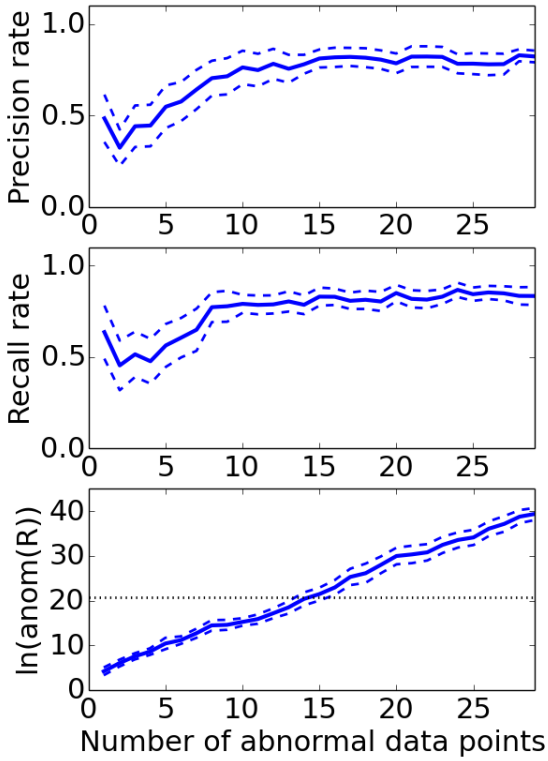
$$\text{precision} = \frac{|\{\boldsymbol{s}_i : \boldsymbol{s}_i \in R \cap S\}|}{|\{\boldsymbol{s}_i : \boldsymbol{s}_i \in R\}|} \tag{3.31}$$

$$\text{recall} = \frac{|\{\boldsymbol{s}_i : \boldsymbol{s}_i \in R \cap S\}|}{|\{\boldsymbol{s}_i : \boldsymbol{s}_i \in S\}|} \tag{3.32}$$

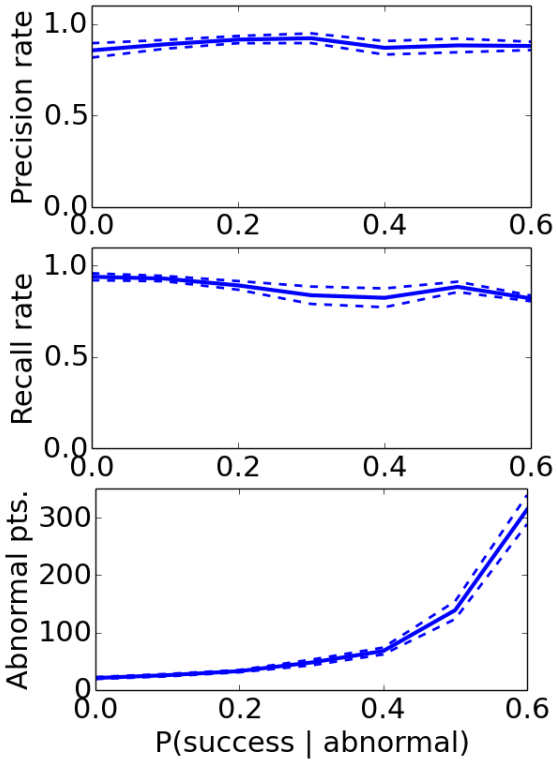
Figure 3.3a show the results of these experiments for $p = 0.8$, $q = 0.2$. With a relatively low number of samples (around 10), precision and recall both get to about 80%, and they keep slowly

increasing as more data arrives. The anomaly value, on the other hand, appears to grow exponentially as more abnormal data arrives ($\ln \mathrm{anom}(R)$ grows linearly), showing that the algorithm quickly gains the necessary confidence to declare that an anomaly occurred. Figure 3.2 shows an example of how an ellipse evolves as more data becomes available. Both $\mathrm{anom}(R)$ and the accuracy of the ellipse approximation increase with the number of abnormal data points.

Note that even when the ellipse approximates the true anomaly very well, as in Figure 3.2c, precision and recall rates still do not reach 1.0. The first reason for this is that the true shape of each region cannot be arbitrarily-well approximated by any ellipse; the efficiency of choosing a relatively small parameterization (i.e., ellipses) comes with the cost of the inability to represent regions arbitrarily well. The other reason why precision and recall do not reach 1.0 is that, if there are missed shots outside of the truly abnormal region, an ellipse that includes those shots has a higher value than an ellipse that does not, and conversely with successful shots that are inside of the abnormal region; thus, *for finite numbers of observations*, the ellipse that maximizes anomaly is not necessarily the one that best matches the true anomaly region.




(a) Precision, Recall and Anomaly value with varying number of abnormal data points




(b) Precision, Recall and Number of abnormal points when threshold anomaly was passed.

Figure 3.3: Synthetic data experiments results. Blue dashed lines indicate standard error bars. The black dashed line is the highest anomaly value observed during nominal execution.

The goal of the second experiment was to determine the effect of the magnitude of the inaccuracy on the algorithm's performance. To do this, we analyzed the number of data points required for the detector to reach $\ln(\mathrm{anom}(R)) = 41.6$ (twice as much as the maximum observed during

normal execution), as the abnormal distribution $q$ approached the normal distribution $p = 0.8$. Figure 3.3b shows the results of this experiment. The number of data points required to be certain of an anomaly appears to increase exponentially as the anomalous distribution gets closer to the normal distribution. This result is expected, as the number should go to infinity as $q \to p$, at which point the distributions are indistinguishable. Furthermore, it is noteworthy that the precision and recall rate stay close to constant for a given anomaly threshold. This suggests that the precision and recall performance of the detector is approximately independent of the magnitude of the failure, given an anomaly threshold $a_{\text{thresh}}$.

### 3.5.2 CoBot motion single RIM detection

We also evaluated FARO by introducing inaccuracies in the CoBot's motion model domain described in Section 1.2.1. We define the motion state of the CoBot by its translational and rotational position and velocities: $\boldsymbol{x} \equiv [x\ y\ \phi\ \dot{x}\ \dot{y}\ \dot{\phi}]^\top$; the observations of execution is $\boldsymbol{z} \equiv [\Delta\dot{x}\ \Delta\dot{y}\ \Delta\dot{v}]$, the vector difference between the CoBot's measured velocity, obtained from its wheel encoders, and its expected velocity, based on its velocity command and its state. These observations, which are execution residuals, are expected to be near zero during nominal execution.

After running the CoBot for 20 minutes of nominal execution, and determining that the highest anomaly value during nominal execution of $\ln(\text{anom}(R)) = 20.1$, we proceeded to inject inaccuracies into its model and test the detector. We introduced four types of RIMs separately:

**Encoder failure** Every time the CoBot moves, one of its wheel encoders observes $(1 - \epsilon)d$, at each timestep, where $d$ is the displacement of the wheel returned during nominal execution. This failure mode evaluates FARO for RIMs that occupy the whole domain.

**Collision** During an otherwise normal run of the robot, a sudden collision happens. This failure mode evaluates FARO for very small, localized RIMs.

**Corridor failure** The wheel encoders fail by returning $(1 - \epsilon)d$, as above, but only in a particular corridor of the building. This failure mode tests FARO for RIMs of medium extent over state-action space. We note that this RIM encompasses a particular sub-region of two of the dimensions of state-action space, and it is global in the others.

**Left turn failure** The robot's execution is nominal except when it turns left (i.e., $\dot{\phi} > 0$), in which case each of its wheels moves only at $(1 - \epsilon)v$ for a velocity command $v$. Since the robot turns only at intersections or when it needs to face a doorway, this failure mode tests the algorithm when small clusters of abnormal points happen infrequently and far apart.

For each of these experiments, the robot was commanded at a high level to navigate to various offices around the building, using its autonomous navigation algorithms [8], and FARO was running constantly. The route the robot took was different each time, as it was simply commanded to go to various rooms. Table 3.1 summarizes the results of running the algorithm under the different failure modes, with $\epsilon = 0.05$.

The algorithm shows high precision and recall for each one of the experiments. The variance in the precision and recall rates is small but not insignificant, indicating that some runs were probably more difficult than others. For example, recall may be higher for a run in which the robot goes down the bad corridor in the same direction multiple times than for a run in which it went down

| RIM | Anomaly Thresh = 40.2 | | | End of experiment | | |
|---|---|---|---|---|---|---|
| | Data Points | Precision | Recall | Anomaly | Precision | Recall |
| Encoder | $58 \pm 8.2$ | $1.0 \pm 0.0$ | $0.76 \pm 0.15$ | $337 \pm 14$ | $1.0 \pm 0.0$ | $0.92 \pm 0.03$ |
| Collision | $4.5 \pm 2.1$ | $1.0 \pm 0.0$ | $0.81 \pm 0.09$ | $1535 \pm 54$ | $1.0 \pm 0.0$ | $0.85 \pm 0.03$ |
| Corridor | $60 \pm 11$ | $0.94 \pm 0.02$ | $0.77 \pm 0.15$ | $199 \pm 28$ | $0.97 \pm 0.01$ | $0.90 \pm 0.02$ |
| Left turn | $31 \pm 5.1$ | $1.0 \pm 0.0$ | $0.79 \pm 0.07$ | $203 \pm 22$ | $0.80 \pm 0.18$ | $0.47 \pm 0.12$ |

Table 3.1: CoBot experiment results. For each experiment, the table shows statistics at the time the anomaly threshold (two times the largest anomaly observed during normal execution) was surpassed, and statistics at the time the robot was stopped.

and up the bad corridor: In the latter case, the ellipse had to expand over a gap between regions with $\phi \approx \alpha$ and those with $\phi \approx \pi + \alpha$, where $\alpha$ is the angle of the corridor itself.

The experiment that stands out in terms of results is the *Bad turn left*, for which the recall rate is significantly lower than the others. This is because the anomaly happens in very disjoint regions of state space (only when the robot needs to turn left), with no data between them. This also explains why the final recall in this failure mode is lower than at the time of detection: more disjoint abnormal regions were visited after the time of detection, and some of them were not joined to the main ellipse. This result shows promise for the potential addition of active exploration to our algorithm: if two disjoint abnormal regions are detected, and there is no data supporting or denying the existence of an anomaly between them, exploring that region would provide evidence one way or the other.

### 3.5.3 Interception-Keepaway robot soccer domain

We evaluate the DMAPS detection algorithm of Section 3.3 and the benefits of model correction as presented in Section 3.4 on a complex sub-problem of the robot soccer domain introduced in Section 1.2.2. The domain, which we denominate Interception-Keepaway, is strongly inspired by the Keepaway domain introduced for 2D robot soccer simulation [88].

#### Interception-Keepaway definition

The keepaway domain is a sub-problem of autonomous robot soccer that has become a benchmark problem in the 2D simulated robot league [87]. In this domain, a team of $n$ *keepers* tries to keep the soccer ball within a bounded 2D region, and away from a team of $m$ *takers*, who try to gain possession of the ball. The task is divided into *episodes*, each beginning with the robots and the ball in particular semi-random positions on the field [87]. An episode ends when the ball leaves the bounded region or it is held by one of the takers for a significant period of time, at which point a new episode begins. Robots are rewarded for each time step an episode persists.

To the best of our knowledge, Keepaway has not been introduced to a real robot domain before. This is a problem that presents several challenges, such as the complex dynamics and noise of the real world. Most important for our purposes, however, is the problem of adapting throughout a single game. Unlike simulation, real robots cannot run millions of trials to approach an optimal

policy. Running real robot trials takes human effort, causes wear on the robots (changing their dynamics in the process), and cannot be sped up. Robots thus need to adapt online and from sparse observations, especially to perform well against unknown opponents in realistic timescales.

For the work in this thesis, we use a modified version of the Keepaway problem, which focuses on passing and preventing interceptions. The modifications are the following:

**Takers behavior:** In Keepaway, 2 takers go to the ball, to try to steal it from the keeper $k_0$ currently holding it, while the remaining $(m-2)$ block possible passes. In our domain, takers focus on ball interception rather than stealing a stationary ball: One taker positions itself between $k_0$ and the most open keeper, at a small distance from $k_0$, while the remaining takers position themselves between $k_0$ and each of its most open teammates $k_i$, at a small distance from $k_i$. Keepers thus have two types of pressure from the takers: close marks on potential receivers and close marks on $k_0$. When the ball is in motion, the taker with the lowest interception time will attempt to intercept it.

**Performance measurement:** Instead of rewarding the duration of an episode, performance is measured by the average completion rate of passes as $\frac{n(\text{success})}{n(\text{success})+n(\text{failure})}$. This performance measure more directly focuses on the problem of passing. For this work, we define a successful pass as one in which the ball touches a keeper before it touches a taker or goes out of bounds, while a failed pass is one in which the ball touches a taker before it touches a keeper or the ball goes out of bounds, but other definitions are possible.

### Model correction for Keepaway

We focus on the passer's decision making problem. Each time a keeper receives the ball, it must choose where to pass next to maintain possession. The physical state of the world $\boldsymbol{s}$ is a vector of size $\dim(\boldsymbol{s}) = 6n + 6m + 4$ containing the 2D translational coordinates $\boldsymbol{l}_i$ and 1D rotational coordinates $\phi_i$ of each robot, their first time derivatives $\boldsymbol{v}_b$, and the ball's 2D location $\boldsymbol{l}_b$ and velocity[4]. The actions available to the robot are the legal velocities $\boldsymbol{v}_b$ ($|\boldsymbol{v}_b| \leq 8\frac{m}{s}$) at which it can kick the ball, discretized by magnitude and direction.

We monitor the expected probability of success of passes, $P(\boldsymbol{z} = 1|\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{\theta}^0)$, where an observation of $\boldsymbol{z} = 1$ means the pass succeeded and $\boldsymbol{z} = 0$ means the pass failed. The nominal model $\boldsymbol{\theta}^0$ is based on the planner's estimate of the time $t_i(\boldsymbol{l}_i, \boldsymbol{v}_i, \boldsymbol{l}_b, \boldsymbol{v}_b)$ that each robot $i$ would need, starting at location $\boldsymbol{l}_i$ with velocity $\boldsymbol{v}_i$, to intercept a moving ball starting at location $\boldsymbol{l}_b$ with velocity $\boldsymbol{v}_b$. Given such an estimate (e.g., [6]), the model computes the keepers' time to ball $t_k$:

$$t_k(\boldsymbol{l}_b, \boldsymbol{v}_b) = \min_{i \in \text{keepers}} t_i(\boldsymbol{l}_i, \boldsymbol{v}_i, \boldsymbol{l}_b, \boldsymbol{v}_b), \tag{3.33}$$

and similarly $t_t$ for takers. We thus define the nominal model $\boldsymbol{\theta}^0$ as:

$$P(\boldsymbol{z}|\boldsymbol{s}, \boldsymbol{a} = \boldsymbol{v}_b, \boldsymbol{\theta}^0) = \Phi\left(\frac{t_t(\boldsymbol{l}_b, \boldsymbol{v}_b) - t_k(\boldsymbol{l}_b, \boldsymbol{v}_b)}{\sigma_t}\right), \tag{3.34}$$

[4]For this work, we have chosen to focus on ground passes, thus the third dimension of the world is ignored; also missing is the ball spin and the robots' internal states.

Where $\Phi$ denotes the standard normal cumulative distribution, and $\sigma_t$ is an uncertainty factor. That is, we model the probability of success as being entirely dependent on which robot has the shortest interception time, plus normal noise.

To conclude the definition of the monitor from Algorithm 1, we must define a feature extraction function $\chi(\boldsymbol{s}, \boldsymbol{a})$ (see line 2). For each expectation $e_i$ about taker robot $i$, we extract a 5-dimensional feature vector:

$$\chi(\boldsymbol{s}, \boldsymbol{a}) = (\boldsymbol{l}'_i, \boldsymbol{v}'_i, |\boldsymbol{v}_b|), \tag{3.35}$$

where $\boldsymbol{l}'_i$ and $\boldsymbol{v}'_i$ are the location and velocity of robot $i$ relative to the ball at the time the pass starts, rotated by the direction of the pass. By excluding features of the keepers, we implicitly assume that the only source of unmodeled behavior is the taker robots.

Algorithm 1 is thus fully defined for Keepaway, and runs each time a pass ends, providing corrections to the planner.

The planning model used here is a simple one, as the focus of this work is performance improvement through execution monitoring, rather than optimal planning. We use a greedy planner that chooses the action that maximizes the expected immediate reward. In Keepaway, that means the passing robot maximizes the expected probability of success of its next pass, and does not plan for multiple passes in the future. In practice, then, the planner always chooses the action that maximizes Equation 3.30, where $\boldsymbol{\mu}_0$ is given by the expectation of Equation 3.34. Section 3.5.4 presents an empirical evaluation of this monitor.

### 3.5.4 Interception-Keepaway multiple RIM detection and correction

To empirically demonstrate our model correction framework, we implemented it on the robot soccer domain of Section 1.2.2. The algorithm was extensively evaluated using a realistic PhysX-based simulator, which employs the same interface to the AI as the real world does, models the robots at the component level, and simulates physics to high detail (e.g., it models the angular momentum imparted on the ball when a robot touches it with its spinning dribbling mechanism). Since we seek to improve high-level robot decisions, rather than low-level controllers, simulation is a particularly useful means of obtaining statistically significant results, which can then be corroborated on the real robots.

#### Evaluation Metrics

The first metric by which we evaluated the model-correction framework is Task Performance (TP). The ultimate goal of our monitor is to improve TP in environments with RIMs, which makes it a natural metric to evaluate our framework. TP was measured by the average pass completion rate of a particular model $\boldsymbol{\theta}$ as:

$$\mathrm{TP}(\boldsymbol{\theta}) = \frac{n(\text{success}|\boldsymbol{\theta})}{n(\text{success}|\boldsymbol{\theta}) + n(\text{failure}|\boldsymbol{\theta})}. \tag{3.36}$$

While TP is an intuitive evaluation metric, it is highly dependent on the task at hand. For example, improving TP by 1% in a task that originally had 50% success rate is much less meaningful

than improving TP in a task that originally had 98% success rate. An evaluation metric that more objectively measures model correction performance is the Failure Reduction Rate (FRR) of the corrected model $\boldsymbol{\theta}^+$ with respect to the baseline model $\boldsymbol{\theta}^0$:

$$\mathrm{FRR}(\boldsymbol{\theta}^+, \boldsymbol{\theta}^0) = 1 - \frac{1 - \mathrm{TP}(\boldsymbol{\theta}^+)}{1 - \mathrm{TP}(\boldsymbol{\theta}^0)}. \tag{3.37}$$

FRR measures the expected percentage of failures that were eliminated by correcting the model. A perfect learner, in a task for which perfect performance is achievable, would eventually reach $\mathrm{FRR} = 1$.

A different metric used for evaluation is Model Prediction Accuracy (MPA). While improvement in TP is the main goal, it does not capture the full success or failure of the framework. We are also interested in evaluating how well the predictions made by the corrected model match the observations made during execution. Modeling the world accurately is desirable, independently of TP, because it enables robots to generalize to different tasks in the same domain. For example, models acquired during Keepaway learning could generalize to passing in the wider problem of full soccer. MPA was measured by the average likelihood of observations given the model used for prediction of that observation:

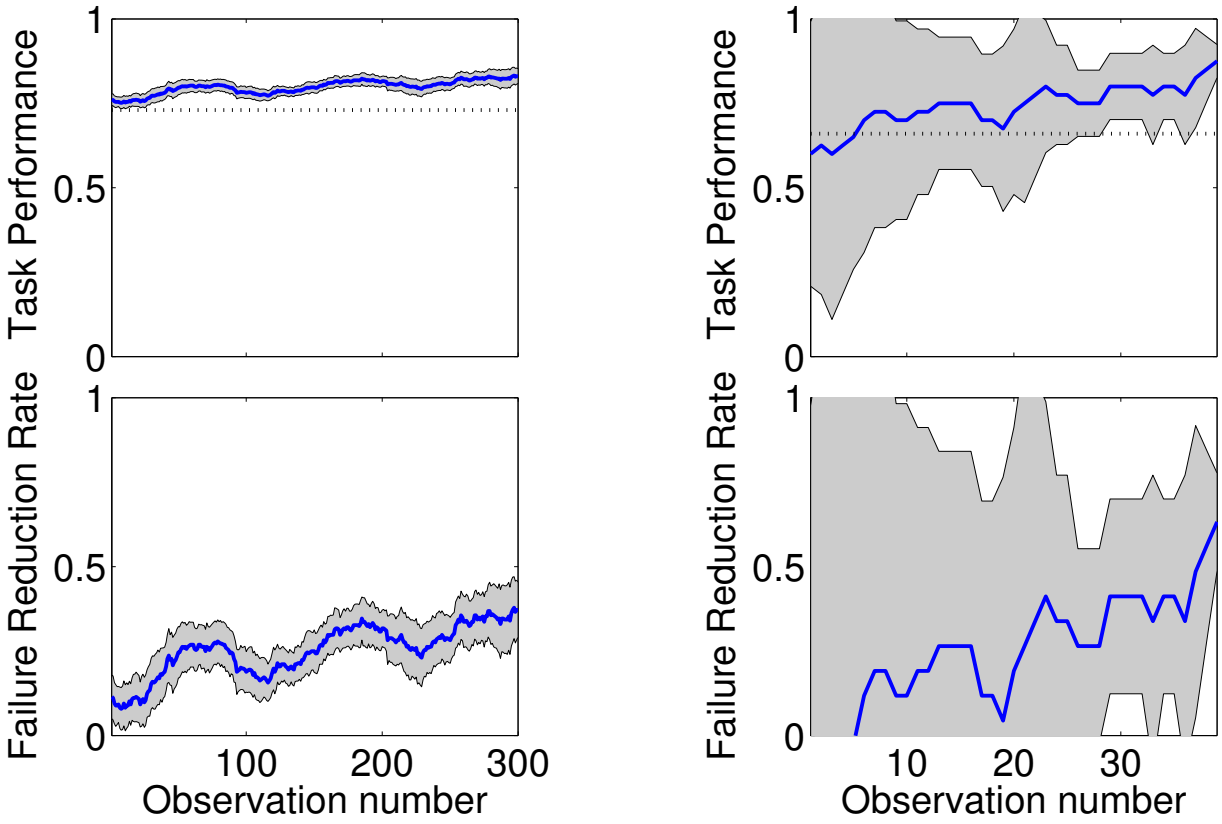
$$\mathrm{MPA}(\boldsymbol{\theta}) = \frac{1}{|\boldsymbol{Z}|} \sum_{(\boldsymbol{x},\boldsymbol{z}) \in \boldsymbol{Z}} P(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}). \tag{3.38}$$

For all of our performance metrics, and throughout our experiments, we use as a baseline the original model $\boldsymbol{\theta}^0$.

When analyzing results, we keep in mind the timescale of the learning process in which we are most interested. A game of robot soccer lasts 20 minutes (1200 seconds). During our keepaway tests, we measured that robots completed, on average, around 0.45 passes per second. Therefore, the upper limit number of passes they could perform in a game is around 540. A considerable portion of that time will be spent not passing (e.g., opponent possession, dead time between plays, shots on goal). However, this estimate lets our timescale of interest lies in the order of $10^2$ passes. The experiments conducted here thus focused on such timescales.

### Experimental Results

First, we conducted extensive simulation tests to determine in a statistically significant way the evolution of $\mathrm{TP}(\boldsymbol{\theta}^+)$ and $\mathrm{MPA}(\boldsymbol{\theta}^+)$ as the monitor acquires new observations. Figure 3.4a shows a moving average (window size 50) measurement of TP and FRR as a function of how many passes the robots have performed, demonstrating an evident performance improvement as the model is corrected with experience. The first noteworthy aspect of this result is that performance quickly improves with the first few observations: the first data point shows $\mathrm{FRR} \approx 0.1$; that is, averaging over the first 50 observations, we see a 10% reduction in failures. Furthermore, as new pass results are observed, performance keeps improving, achieving a failure reduction of about 40% within the first few hundred passes. Conducting less extensive experiments on the real robots showed




(a) Simulation

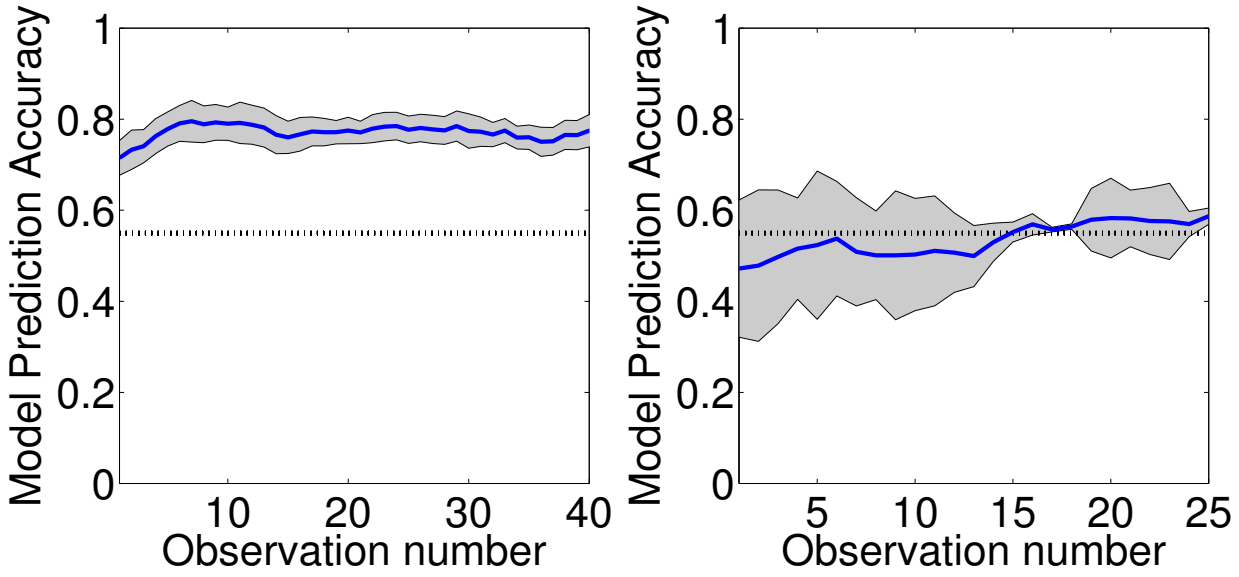
(b) Real robots

Figure 3.4: Moving average of passing performance evaluation as a function of number of passes performed. The shaded area shows the 95% confidence margin; the dotted black horizontal line indicates average baseline performance.

a similar trend, shown in Figure 3.4b. Baseline and adaptation performance were both higher in simulation than in the real robots, due to the complications added by the real world.

To evaluate MPA adequately, we ran experiments in which the robots ignored model corrections suggested by the execution monitor. This allowed us to evaluate the predictive performance of the monitor without the confounding factor of altered robot behavior. We evaluate MPA exclusively for points inside of detected RIMs, as this is where model improvement is expected to occur due to our monitor. For points that lie outside of the detected RIMs, the model (and thus MPA) remains unchanged by the monitor.

Figure 3.5a shows $\mathrm{MPA}(\boldsymbol{\theta}^+)$ evaluation for simulation. With model correction, observations show a significantly better fit to the model than without correction. The ideal $\mathrm{MPA} = 1$ is only achievable by distributions with 0 variance, which is certainly not the case in our Keepaway domain. The realistic goal of the monitor is not to reach $\mathrm{MPA} = 1$, but to show significant improvement over the initial global model. Figure 3.5b shows less extensive real-robot test results, which show a similar trend as simulation.




(a) Simulation

(b) Real Robots

Figure 3.5: Moving average of prediction accuracy $\mathrm{MPA}(\boldsymbol{\theta}^+)$, as a function of observations inside of RIMs. The shaded area shows the 95% confidence margin; the dotted black horizontal line indicates average baseline $\mathrm{MPA}(\boldsymbol{\theta}^0)$

These results support the efficacy of our framework for short-term task performance and prediction accuracy improvement. Anecdotal evidence from this domain also suggests a different benefit of our approach: detected RIMs may have attached semantic explanation that may be beneficial to system designers, or perhaps eventually to the robots themselves. After conducting the experiments above, the model designers found that the discovered RIMs corresponded to flaws in the design of their simple prediction model of Equation 3.34. Some RIMs corresponded to inadequacies in the computation of navigation times $t_k$ and $t_t$, while others corresponded to inad-

equacies in the assumption that the probability of success corresponds exclusively to a smoothly varying function of $t_t - t_k$. While we did not focus on the problem of assigning semantic meaning to discovered anomalous subspaces, it is an area of interest for future research.

## 3.6 Chapter summary

This chapter presents the technical detail of this thesis' approach to detecting context-dependent model inaccuracies in low-dimensional planning spaces. The approach consists of enabling the robot to autonomously search for the parametric region of state-action space in which observed outcomes deviate most significantly from the robot's nominal model, and then deciding whether such deviation is statistically significant.

Section 3.1 sets up the necessary background material to develop the detection algorithm. First, the section describes an inaccuracy measure that reflects the level of anomaly of the robot's behavior in a particular region of its state-action space; the robot searches for the region that maximizes this inaccuracy measure. The section also presents the space over which the robot searches for anomalies: a set of parametric regions.

Section 3.2 presents this thesis' algorithm for detecting a single parametric Region of Inaccurate Modeling (RIM). At each time step, the robot runs an optimization to find the parametric region that maximizes the aforementioned inaccuracy value. Once the optimization finds this maximum anomaly region, a statistical test determines whether the detected region is likely to constitute an anomaly.

Section 3.3 extends the notion of RIM-detection to domains with multiple model inaccuracies. First, the section shows a derivation of a measure of inaccuracy for multiple regions of state-action space. Then, the section presents an algorithm for effective detection of multiple RIMs.

Section 3.4 shows how the robot can use RIM-detection to improve task performance by applying corrections to its planning models. The robot creates a new planning model that is a mixture model combining its original nominal model with newly discovered behavioral modes that correspond to each of the detected RIMs.

Section 3.5 demonstrates empirically the value of this chapter's technical contributions. The RIM-detection algorithms successfully detect model inaccuracies in a variety of domains: the CoBot mobile service robot, the CMDragons team of soccer-playing robots, and a synthetic golf-robot domain. Furthermore, in the CMDragons domain, the robots were able to significantly improve their performance online in a sub-problem of the robot soccer problem.

# Chapter 4

# Detection of RIMs in High-Dimensional Spaces

This chapter contributes an approach for RIM-detection that scales well to high-dimensional domains. While Chapter 3 presents an approach that effectively detects RIMs in low-dimensional domains online, the performance of this approach does not scale well to higher-dimensional spaces, as the experiments in Section 4.2 show. Since most real robot domains have significantly higher dimensionality than the experiments of Chapter 3, we seek to enable the robot to autonomously find RIMs in such spaces.

**Key assumption: Low-dimensional RIMs.** To find RIMs in high-dimensional context spaces efficiently and effectively, we assume that the RIMs can be fully described in a low-dimensional projection of the robot's observations, although the correct projection is unknown a priori. For example, the golfing robot's performance may be affected by an unseen bump on the field (2D RIM), or by the wind velocity vector (3D RIM), or even by lighting conditions generated by the sun at a particular time of the day, during some months of the year in a particular section of the field (4D RIM); but not by a RIM that is intrinsically high-dimensional. In practice, this assumption is met by every real-world model inaccuracy our robots have encountered. The challenge of the problem, then, lies in identifying the best low-dimensional projection efficiently to find the RIMs in that subspace.

**Feature Selector for RIM-detection.** To find the correct low-dimensional projection, we contribute a Feature Selector for RIM-detection (FS-RIM) approach. This approach leverages the work on RIM-detection in low-dimensional spaces of Chapter 3 to iteratively (a) conduct a heuristic best-first search for subsets of features likely to contain RIMs, and (b) explicitly search for RIMs in the most promising projections. The key to conducting this search effectively is to define heuristics that informatively select projections likely to contain RIMs, without incurring in the full cost of searching for RIMs in those projections.

**Evaluation and results.** We evaluate the effectiveness of FS-RIM in two domains: simulated data from a high-dimensional version of the golf-putting robot scenario, and real motion data from the CoBot mobile service robots [94]. Results show that the robots are able to autonomously detect various types of injected model inaccuracies effectively and efficiently, significantly outperforming the RIM-detection algorithms of Chapter 3 as domain dimensionality increases.

**Chapter organization.** Section 4.1 presents the technical detail of our search-based feature selection algorithm, while Section 4.2 validates this approach through experiments on the golf-putting robot, CoBot, and spacecraft landing domains.

## 4.1 Feature selection algorithm

This section presents the technical details of the main contribution of this chapter: a Feature Selector for RIM-detection (FS-RIM) in high-dimensional domains. Similarly to related work [64], the goal is to find the region $R^+$ of context space that maximizes the anomaly measure $\mathrm{anom}(R)$ of Equation 2.5. The key assumption that enables FS-RIM to find the maximum anomaly region $R^+$ efficiently is that these RIMs are intrinsically low-dimensional regions embedded in a high-dimensional context space. This enables the use of a search-based Feature Selection algorithm to greatly reduce the dimensionality of the search for RIMs.

### Wrapper-style feature selection

The FS-RIM feature selector is a wrapper-style algorithm [50], in which the optimal projection of context space is found by evaluating the function to be maximized –i.e., by finding $R^+$– in selected low-dimensional projections of the full context space. Choosing a wrapper approach enables the approach to leverage previously existing low-dimensional approaches (e.g., exhaustive search [53] or the FARO optimization-based search [64]) in its search for high-dimensional RIMs. Thus, the approach assumes that there exists an function, called $\mathrm{findRIM}(\boldsymbol{Z}, \boldsymbol{\theta}^0)$, which, given a set of low-dimensional contextual observations $\boldsymbol{Z}$ and a model of nominal behavior $\boldsymbol{\theta}^0$, can find the region $R^+$ that maximizes Equation 2.5.

### Feature selection as best-first search

Since there are $2^{|\mathcal{F}|}$ different possible subsets of the full set of features $\mathcal{F}$ in the robot's context space, the core problem is to efficiently and effectively search through the space of possible projections: We want an algorithm that explores more informative projections –i.e., those that are likely to contain high anomaly regions– before exploring less informative ones –i.e., those less likely to contain high anomaly regions. FS-RIM uses an informed best-first search over elements $F \in 2^{\mathcal{F}}$ of the power set $2^{\mathcal{F}}$ of $\mathcal{F}$. The search is conducted on a graph $G$ in which the vertices $V(G)$ of the graph are the possible feature sets $F \in 2^{\mathcal{F}}$, and edges $E(G)$ connect a vertex $v_0 = F$ to a vertex $v_1 = F \cup \{f\}$ if $v_1$ is the result of adding a single feature to $v_0$:

$$V(G) = 2^{\mathcal{F}} \tag{4.1}$$

$$E(G) = \{(F, F \cup \{f\}) : F \in 2^{\mathcal{F}}, f \in \mathcal{F}, f \notin F\} \tag{4.2}$$

Figure 4.1 shows an abbreviated illustration of the first three levels of the resulting search tree, which always starts with the empty set of features as the root.

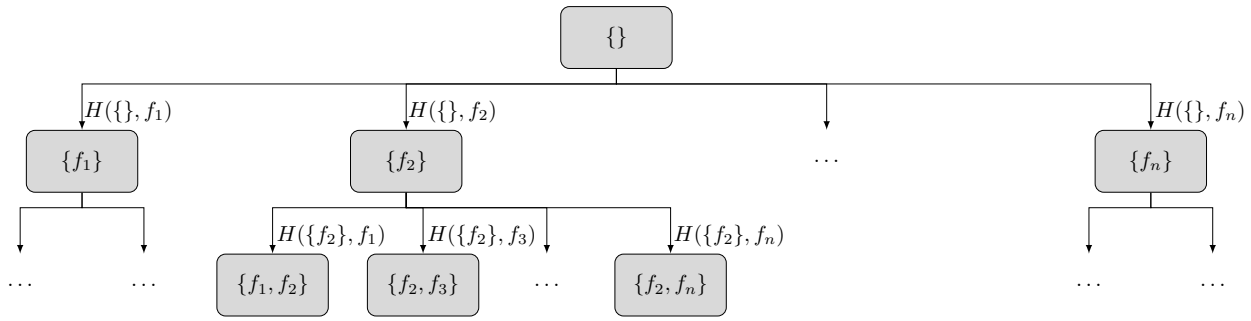



Figure 4.1: Search tree for feature selection. The algorithm always starts the search with no features, and at each step searches the node on the boundary of unvisited nodes with the maximum heuristic value $H$.

### Algorithm description

Algorithm 4 details the search procedure. The search starts by setting the boundary $\mathcal{O}$ of the search to the empty set of features (line 3). For each step $i$ of the search, the algorithm finds the edges that lie at the boundary of the search (line 5) and determines which one to explore using a *heuristic value function* $H(e)$ (line 6). To explore the chosen edge $(F_i^-, f_i)$, the algorithm first projects the original contextual observations $\boldsymbol{Z}$ onto the space spanned by the union $F_i = F_i^- \cup \{f_i\}$ (line 8). Then, the algorithm uses a low-dimensional search method $\mathrm{findRIM}$ to search for the most anomalous region $R^+$ in the resulting low-dimensional space (line 9). While this low-dimensional search method can be one of many options –e.g., exhaustive search if the low-dimensional space is small enough [53]–, here we use the FARO algorithm of Chapter 3. Once the search has completed, and the most likely RIM $R^+$ has been found, the algorithm decides whether the evidence is strong enough to declare a significant model inaccuracy, or not (lines 14-18).

### Algorithm parameters

Line 4 in Algorithm 4 specifies that the search continues until a domain-dependent maximum number of nodes $i_{\max}$ has been expanded. Depending on the requirements of the domain, this search-ending constraint may be exchanged by a time limit instead of a maximum number of expanded nodes. In general, the algorithm does not have a specific stopping condition for its search, but its performance may only improve with execution time or maximum number of nodes $i_{\max}$. If no model inaccuracy is detected during the allotted maximum number of iterations or maximum time, the algorithm returns no inaccuracy.

The threshold $a_{\mathrm{thresh}}$ in line 14 is domain-specific, and can be computed to correspond to a desired rate of False Positive detections. As specified in Section 3.2, an approximate map from threshold to False Positive rate can be computed empirically through simulations of the domain under nominal execution –for example, to determine the anomaly threshold for a False Positive rate of 5%, one may run 100 simulations of the domain under nominal execution, run FS-RIM each time, and then set the threshold $a_{\mathrm{thresh}}$ to the value of the fifth most anomalous region detected during nominal execution. In our experiments, we have used this 5% False Positive rate.

**Algorithm 4** Algorithm for detection of a RIM in a high-dimensional context space.
*Input:* List of contextual observations $\boldsymbol{Z}$, nominal behavior model $\boldsymbol{\theta}^0$.
*Output:* A RIM $R^+$, or $\emptyset$ if no RIM is detected.

**function** FS-RIM( $\boldsymbol{Z} = [(\boldsymbol{x}_0, \boldsymbol{z}_0), \ldots, (\boldsymbol{x}_n, \boldsymbol{z}_n)]$, $\boldsymbol{\theta}^0$)
  $R^+ \leftarrow \emptyset$ ▷ Most anomalous region thus far
  $\mathcal{O} \leftarrow \{\emptyset\}$ ▷ Graph search boundary
  **for** $i = [0, 1, \ldots, i_{\max}]$ **do**
    $E_i \leftarrow \{(F \in \mathcal{O}, f \in \mathcal{F}) : f \notin F\}$
    $(F_i^-, f_i) \leftarrow \arg\max_{e \in E_i}[H(e)]$
    $F_i \leftarrow F_i^- \cup \{f_i\}$
    $\boldsymbol{Z}_i \leftarrow [(\boldsymbol{x}_t^i, \boldsymbol{z}_t) : \boldsymbol{x}_t^i = \mathrm{Project}(\boldsymbol{x}_\mathrm{t}, \mathrm{F_i})]$
    $R_i^+ \leftarrow \mathrm{findRIM}(\boldsymbol{Z}_i, \boldsymbol{\theta}^0)$
    **if** $\mathrm{anom}(R_i^+) > \mathrm{anom}(R^+)$ **then**
      $R^+ \leftarrow R_i$
    **end if**
  **end for**
  **if** $\mathrm{anom}(R^+) > a_{\mathrm{thresh}}(\boldsymbol{Z}, \boldsymbol{\theta}^0)$ **then**
    **return** $R^+$
  **else**
    **return** $\emptyset$
  **end if**
**end function**

### 4.1.1 Search heuristic

A crucial step in Algorithm 4 is the choice of heuristic function $H(e)$ in line 6. Computation of this heuristic must be relatively efficient in comparison to the $\mathrm{findRIM}$ function of line 9, since the former is invoked for each of the edges in the boundary of the search to decide which one is next explored with the latter. Furthermore, the heuristic must be as informative as possible for RIM-detection –i.e., it should approximate the anomaly value $\mathrm{anom}(R^+)$ of the maximum-anomaly region $R^+$ in that projection.

To compute efficient and informative heuristics for each edge $(F, f)$, FS-RIM leverages the fact that the maximum anomaly region for the projections defined by $F$ has been computed precisely in earlier stages of the search, and that the maximum anomaly region for all the individual features $f$ can be computed only once in $O(|\mathcal{F}|)$ time. Thus, the heuristics can use the corresponding maximum anomaly regions $R^+_F$ and $R^+_{\{f\}}$ in their computations with little extra cost.

We note that our feature selection algorithm's performance relies on the informativeness of these heuristics, which in turn relies on the informativeness of finding the maximum anomaly region $R^+_F$ in a lower-dimensional subspace. In some degenerate cases, a model inaccuracy may be completely hidden in each lower-dimensional space $F$, and only becomes perceptible when the full subspace in which it lives is analyzed. Our algorithm is not able to effectively find the right subspace in such degenerate cases.

For visualization purposes, we describe each heuristic using as an example an edge in which $F = \{f_1\}$ and $f = f_2$, shown in Figure 4.2. However, we note that $F$ is a set that can contain zero or more features, while $f$ is a single feature to be added to $F$. Thus, even though both $R^+_F$ and $R^+_{\{f\}}$ appear as ranges along a single dimension in Figure 4.2, more generally, $R^+_F$ is a $|F|$-dimensional parametric region (ellipsoid in this work), while $R^+_{\{f\}}$ is a one-dimensional parametric region. We note that neither $R^+_{\{f\}}$ nor $R^+_F$ in Figure 4.2 contains all of the missed shots; this is because in their respective 1D projections, extending the region to contain every missed shot would also require containing many more scored shots, thus lowering the overall anomaly value $\mathrm{anom}$ of the region.

#### Anomaly sum heuristic ($H_1$)

The first heuristic, computable in constant time, is given by the sum of the anomaly values of $F$ and $f$:

$$H_1(e) = \mathrm{anom}\left(R^+_F\right) + \mathrm{anom}\left(R^+_{\{f\}}\right). \tag{4.3}$$

Figure 4.3a illustrates the meaning of this heuristic in the case where $F$ is a one-element set: Heuristic $H_1$ is given by the sum of the anomaly values of each of the two regions independently.

For each feature $f$, the maximum anomaly region $R^+_{\{f\}}$ needs to be computed only once at the beginning of the search, so its cost per edge of the search is constant. Furthermore, since the graph vertex containing $F$ has already been explored, $\mathrm{anom}(R^+_F)$ has already been computed by the time edge $e$ is explored (see line 9 of Algorithm 4). This heuristic is very efficient, but may not be extremely informative in domains in which $F$ and $f$ may not seem highly anomalous independently, but they are together.

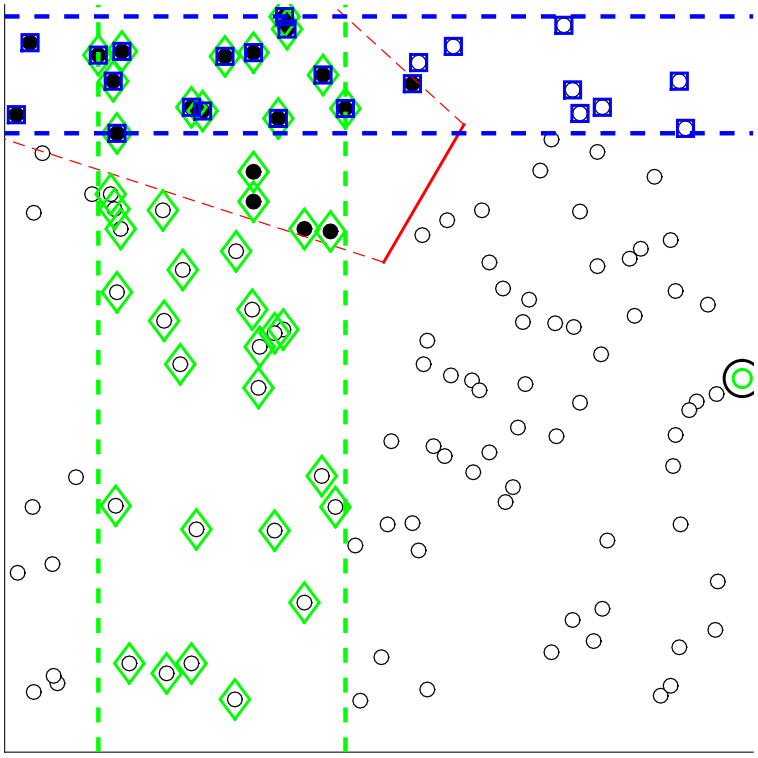
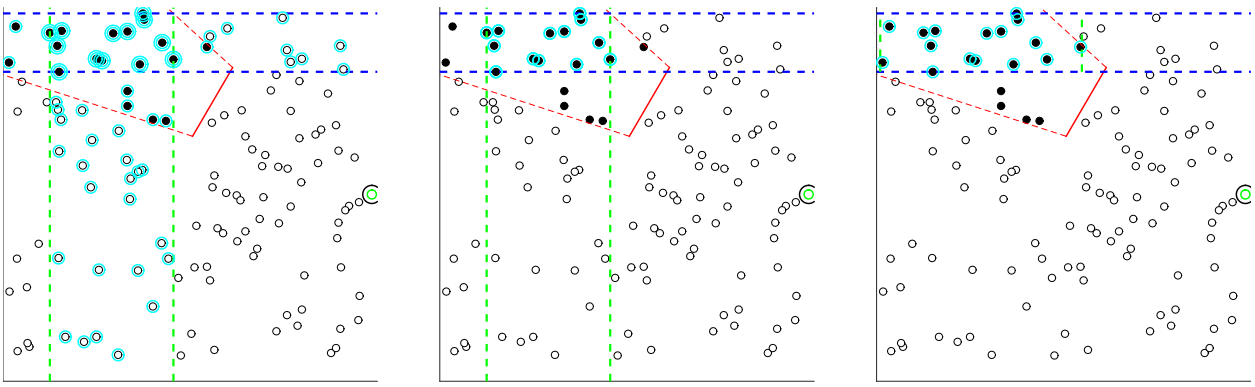


Figure 4.2: Non-sutble golf domain RIM for heuristic visualization: every shot from within the RIM (red lines) is missed (black circles) while every shot from outside the RIM is scored (white circles). Blue dashed lines and squares show the maximum anomaly region when projecting onto $F = \{f_1\}$, while green dashed lines and diamonds show the maximum anomaly region along when projecting onto $f = f_2$.



(a) Heuristic $H_1$ (b) Heuristic $H_2$ (c) Heuristic $H_3$

Figure 4.3: Visualization of the presented heuristics. Blue and green dashed lines show the relevant regions in lower-dimensions, while cyan-highlighted data points are those that contribute to the the heuristic value.

**Region intersection heuristic ($H_2$)**

The second heuristic is more informative than $H_1$, but has a $O(n)$ computational cost per edge, where $n$ is the number of data points. Given edge $e = (F, f)$, This heuristic is obtained by intersecting the points contained in the maximum anomaly region $R^+_F$ from the subspace of features $F$, with those contained in the maximum anomaly region $R^+_{\{f\}}$ of the single-dimensional space of $f$, as illustrated in Figure 4.3b:

$$H_2(e) = \text{anom}\left(R^+_F \cap R^+_{\{f\}}\right). \tag{4.4}$$

In this work, where we use ellipsoids as our chosen parametric regions for optimization, this heuristic computes the anomaly value of the hyper-cylindrical region obtained from intersecting $R^+_F$ and $R^+_{\{f\}}$: $R^+_F$ forms the elliptical base in $F$, while $R^+_{\{f\}}$ constrains the points to those in a particular range along $f$. To compute $H_2$, each point in $R^+_F$ is tested for belonging to the range $R^+_{\{f\}}$, leading to a $O(n)$ computation for each edge.

**Conditional range heuristic ($H_3$)**

Finally, we present a highly informative heuristic with a $O(n^2)$ computational cost. Given edge $e = (F, f)$, $H_3$ computes the precise most anomalous range along dimension $f$, given only the observations contained in $R^+_F$, as illustrated in Figure 4.3c:

$$H_3(e) = \text{anom}\left(R^+_{\{f\}}|R^+_F\right). \tag{4.5}$$

Similarly to $H_2$, this heuristic computes the anomaly value of a hyper-cylindrical region with base $R^+_F$. However, this region is the most anomalous such hyper-cylinder, and thus $H_3$ dominates $H_2$.

Heuristic $H_3$ can be computed in $O(n^2)$ because the most anomalous range $R^+_{\{f\}}$ along a single dimension $f$ can be computed exactly in $O(n^2)$ using dynamic programming, as explained in Appendix A. The same procedure can be used to compute $R^+_{\{f\}}|R^+_F$, using only points within region $R^+_F$. In cases in which the number of points $n$ is prohibitively high, computational costs can be diminished through the use of approximate methods for finding $R^+_{\{f\}}$ (e.g. [64]) or by binning points along feature $f$.

## 4.2 Empirical evaluation

The performance of the contributed Feature Selector for RIM-detection (FS-RIM) was evaluated via experiments on simulated data from the golf-putting domain, on real robot data from the CoBot domain, and on unknown anomalies from the NASA spacecraft landing simulation. The primary purpose of this experimental validation is to demonstrate a significant performance improvement of RIM-detection algorithms in high-dimensional domains using FS-RIM, when compared to the FARO method without feature selection, and when compared to non-informed search for Feature Selection. Additionally, the experiments provide a comparison among the different heuristic functions of Section 4.1.1.

### 4.2.1 Experimental conditions

The experiments compare the performance of FS-RIM to that of RIM-detection using no Feature Selection, and to RIM-detection using uninformed search as its method for Feature Selection.

**Feature Selection for RIM-detection (FS $\mathbf{H}_i$).** This condition is the Feature Selection algorithm presented in this chapter, with the various heuristics of Section 4.1.1.

**No Feature Selection (No FS).** This condition is exactly the FARO RIM-detection algorithm of Chapter 3. We expect this algorithm to perform comparably to FS-RIM in low-dimensional domains, but to perform substantially worse than FS-RIM in higher-dimensional domains.

**Breadth-First Search Feature Selection (BFS).** This condition uses the Breadth-First Search uninformed search for feature selection, to achieve the same goal as the informed search of FS-RIM. We expect this approach to perform comparably to FS-RIM when the ground truth RIM is implicitly one-dimensional, since FS-RIM begins with a thorough search of the one-dimensional subspaces of the domain, and in domains in which the proposed heuristics for FS-RIM are not an informative descriptor of higher-dimensional RIMs. We expect this approach to perform less well than FS-RIM in domains with multi-dimensional RIMs.

### 4.2.2 Evaluation metrics

The primary performance metric of FS-RIM is its ability to correctly identify data points that lie within a RIM. This is achieved by comparing each point's belonging to the ground truth RIM $R_{\mathrm{GT}}$ to its belonging to the maximum anomaly RIM $R^+$ detected by FS-RIM, if any exists. For a given experiment, then, the number of True Positives ($\mathrm{TP}$), False Positives ($\mathrm{FP}$), True Negatives ($\mathrm{TN}$) and False Negatives ($\mathrm{FN}$) are given by:

$$\begin{aligned}
\mathrm{TP} &= \sum_{(\boldsymbol{x}_i,\boldsymbol{z}_i)\in\boldsymbol{Z}} \mathbb{1}(\boldsymbol{x}_i \in R^+ \wedge \boldsymbol{x}_i \in R_{\mathrm{GT}}) \\
\mathrm{FP} &= \sum_{(\boldsymbol{x}_i,\boldsymbol{z}_i)\in\boldsymbol{Z}} \mathbb{1}(\boldsymbol{x}_i \in R^+ \wedge \boldsymbol{x}_i \notin R_{\mathrm{GT}}) \\
\mathrm{TN} &= \sum_{(\boldsymbol{x}_i,\boldsymbol{z}_i)\in\boldsymbol{Z}} \mathbb{1}(\boldsymbol{x}_i \notin R^+ \wedge \boldsymbol{x}_i \notin R_{\mathrm{GT}}) \\
\mathrm{FN} &= \sum_{(\boldsymbol{x}_i,\boldsymbol{z}_i)\in\boldsymbol{Z}} \mathbb{1}(\boldsymbol{x}_i \notin R^+ \wedge \boldsymbol{x}_i \in R_{\mathrm{GT}})
\end{aligned} \tag{4.6}$$

These measures are combined into a single standard $F_1$ performance metric, which evenly weights the precision and recall of the evaluated algorithms:

$$F_1 = \frac{2\,\mathrm{TP}}{2\,\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{4.7}$$

In particular, the performance of different detection algorithms is evaluated as a function of domain dimensionality. We hypothesize that the performance of FS-RIM would be comparable to FARO with no feature selection in lower-dimensional domains, but we expect to see a significant difference in higher-dimensional domains.

Furthermore, we evaluate the performance of FS-RIM as a function of computational running time. In all experimental conditions, the performance is expected to improve as the algorithm runs for longer. In FS-RIM and BFS, this expected improvement is due to the search being able to expand more nodes, while in FARO it is due to the optimization being able to explore more of the optimization space. In both cases, we also expect the performance to plateau at a certain point in time, once the best option (locally best for FARO) has been found. Evaluating performance as a function of time is essential because the best-first search method presented here has the goal of expanding nodes in an efficient order to avoid having to intractably search the entire space.

### 4.2.3 Golf-putting experiments

The first experimental domain is a variant of the golf-putting domain explained in Chapter 1. While the binary-reward golf domain is useful for explanation and has been explored empirically in previous work [64], here we explore empirically a continuous-reward variant: Instead of a binary reward of 0 or 1, the robot receives a continuous reward proportional to how close to the target the shot ends. This variant enables the work to focus on models defined as Gaussian distributions throughout; however, similar mathematical derivations can be used for other types of distributions.

#### Nominal behavior model

The golf-putting domain was set up as a highly-controlled simulation, as an initial evaluation of FS-RIM with fully known ground truth. In this simulation, the locations $p_i$ from which the robot shoots are chosen uniformly and randomly throughout the field. By design, the robot has a single action to shoot in the known direction of the target, and receives a noisy reward $r_i$ depending on $p_i$:

$$r_i = \bar{r}(p_i) + \epsilon \tag{4.8}$$

where $\epsilon \sim \mathcal{N}(0, 0.1^2)$ is a normally-distributed noise term, and the expected reward $\bar{r}$ at location $p_i$ is given by a linearly-decreasing function of the distance $d_i$ from $p_i$ to the target: $\bar{r}(p_i) = 1.0 - 0.5\frac{d_i}{\text{FieldLength}}$. During training, the robot has access to the expected reward function $\bar{r}(p_i)$, and the parameters of the noise $\epsilon$ are extracted from the data.

#### Context-dependent model inaccuracy

A straight bump, like that of Figure 1.2 is placed randomly on the field, in a different location for each experiment. The bump is always perpendicular to the target on the right, at a minimum distance of $5\%$ of the field and a maximum distance of $75\%$ of the field from the target. The center of the bump is placed at an angle between $135\,\text{deg}$ and $235\,\text{deg}$ from the target, and its subtended angle varies from $\frac{\pi}{8}$ to $\frac{\pi}{7}$ radians. When the robot takes a shot from a location $p_i$ behind the bump, its expected reward is $\hat{r}(p_i) = \bar{r}(p_i) - 0.2$ instead of the original $\bar{r}(p_i)$, creating region behind the

bump in which the robot's model is inaccurate. For each of the experiments below, the simulated robot repeteadly took shots until it had shot 30 times from behind the bump.

#### High dimensional context

Additionally to the two dimensions defining the spatial golf field dimensions, higher context dimensions were introduced as required for each experiment. The value of each data point in each of the added dimension is uniformly distributed in the range $[-1, +1]$. The desired outcome of this experiment, then, is for FS-RIM to be able to distinguish the two features that affect the model inaccuracy –i.e., the x and y spatial dimensions– from the remaining dimensions, which are irrelevant to how well the model predicts the robot's reward.

#### Experimental results

Figure 4.4 shows an example of the detected RIM in a 100-dimensional domain, using FS-RIM with heuristic $H_3$. The algorithm correctly identifies that the shown projection is the most informative one, and proceeds to run an optimization over possible ellipses to find the one most likely to be a RIM.

Figure 4.5 shows the performance of FS-RIM using the three different heuristics of Section 4.1.1 (FS $H_i$), as well as that of the original FARO algorithm [64] without Feature Selection (No-FS) and the uninformed Breadth-First Search (BFS) approach to Feature Selection.

The first result is that the performance of No-FS quickly degrades with the increasing dimensionality of the domain. In a 2D environment, No-FS reaches peak performance before the FS methods since it does not need to initially compute the heuristic values for each feature. However, this small time advantage is overshadowed by the performance deficiency in higher dimensions.

The FS-RIM approach also significantly outperforms the uninformed BFS approach, especially as the dimensionality of the domain grows. The heuristics of FS-RIM guide the search to explore the right two-dimensional subspace much more quickly than BFS, which expands the nodes in an order that does not correlate to anomaly value.

Figure 4.5 also shows that the performance of the FS algorithms scales well with dimensionality. The time required to reach peak performance for each of the heuristics changes from about 18 seconds in the 2D domain, to between 20 and 30 seconds in the 100D domain. Furthermore, the performance of the algorithms, especially for heuristic $H_3$, does not degrade greatly between the 2D and the 100D domains.

### 4.2.4 CoBot experiments

We are interested in monitoring the motion model of the CoBot robots, very similarly to Section 3.5.2. However, while Section 3.5.2 assumed a relatively low-dimensional domain (6D), this evaluation will focus on a high-dimensional state-action space. There is a *high-dimensional space* of contextual variables that may affect the robot's performance, such as the robot's position, velocity and orientation, the presence of humans, the time of the day or day of the week, the presence of

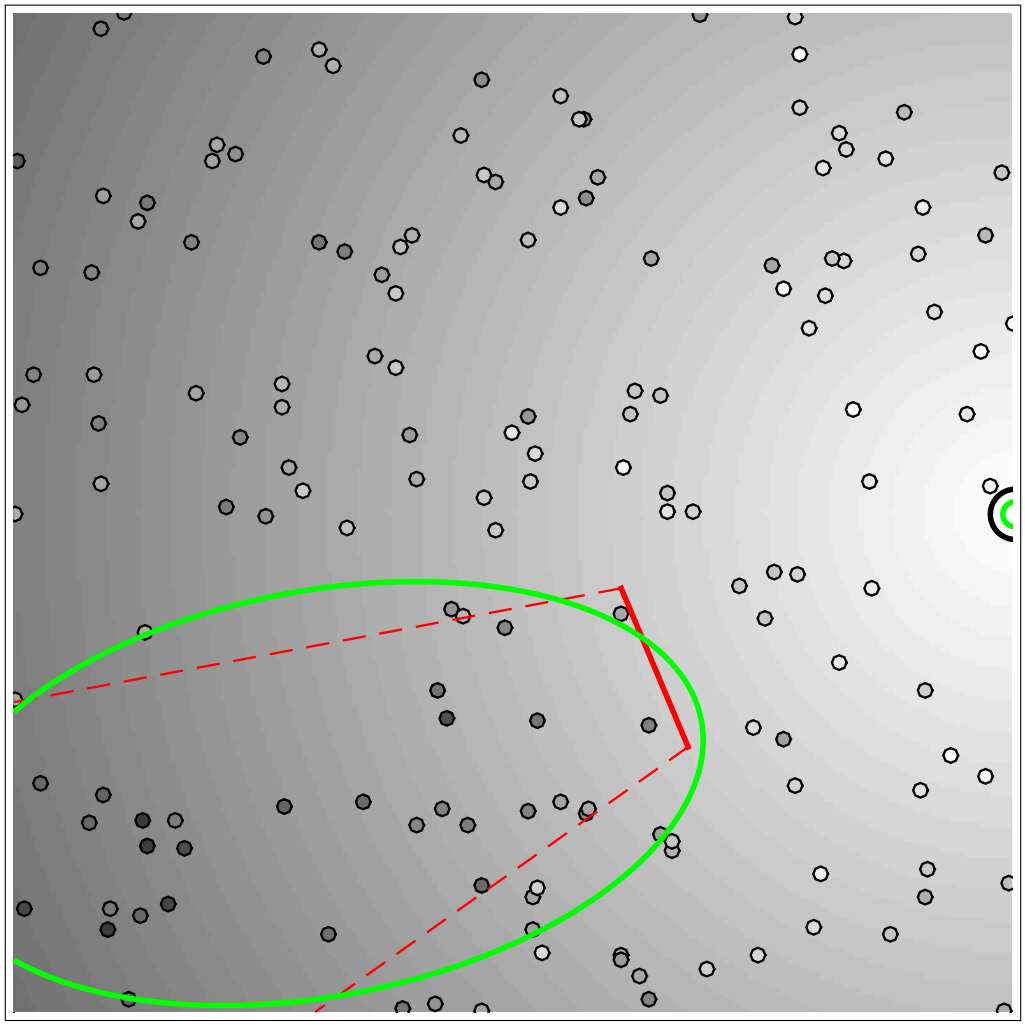
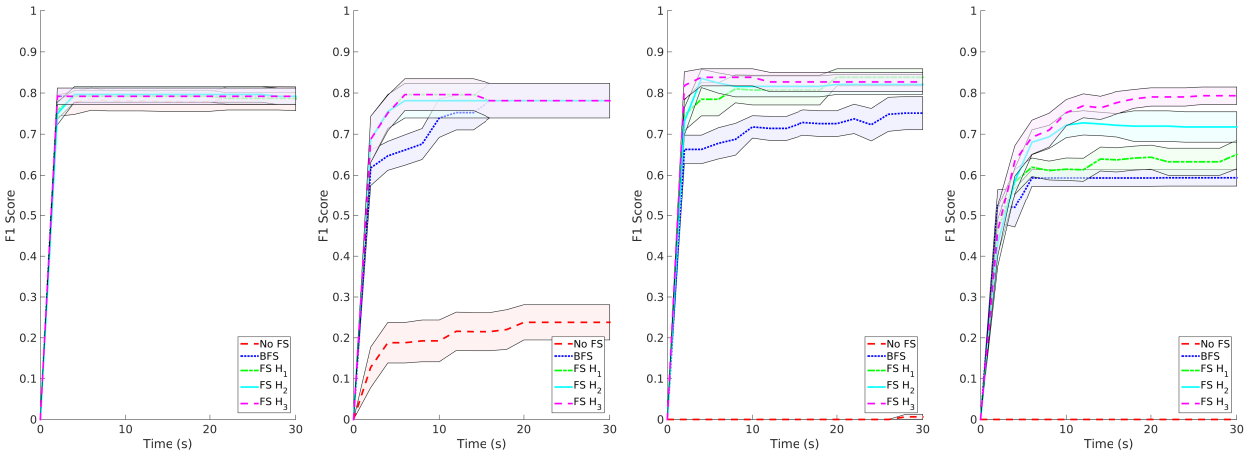


Figure 4.4: Example of FS-RIM applied to a 100-dimensional golf domain. The green ellipse shows the detected RIM, while the red straight lines surround the ground truth RIM. The intensity of each point shows the received reward, while the background intensity shows the expected reward throughout the domain.




(a) 2 Dimensions (b) 4 Dimensions (c) 8 Dimensions (d) 100 Dimensions

Figure 4.5: Detection performance of FS-RIM with different heuristics (FS $H_i$), no Feature Selection (No FS), and Breadth-First Search Feature Selection (BFS) as a function of algorithm running time, in the simulated Golf-putting domain. Shaded areas show a standard error above and below the mean.

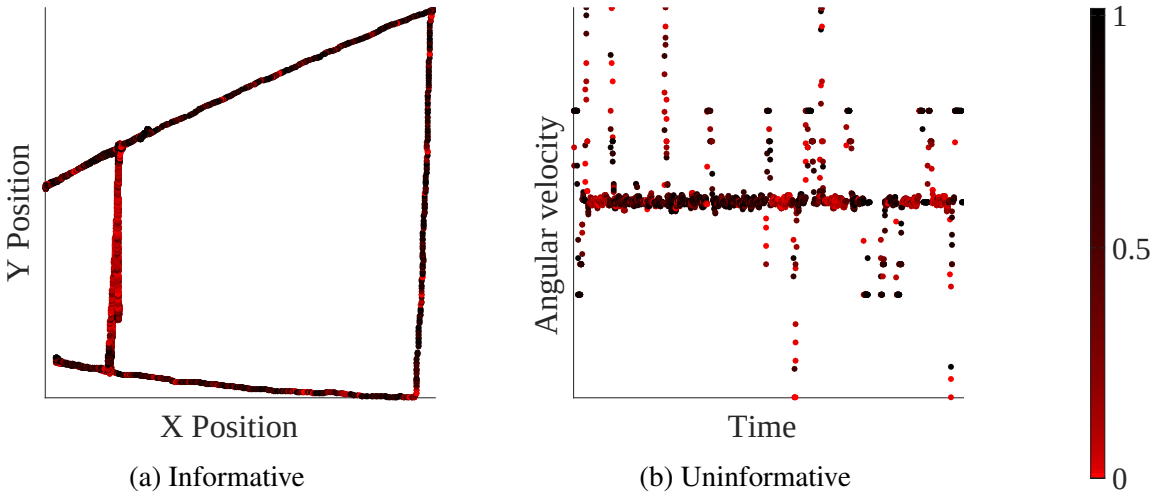
obstacles in its path, its battery voltage, the amount of sun that shines into its sensors, among many others. This section examines model inaccuracies that affect different subsets of these features.

Projecting the data onto informative subsets of features can be crucial for finding RIMs of the CoBot's space. For example, Figure 4.6 shows data in which the CoBot's motion is subtly inaccurate in a particular corridor of the building; projecting the data onto the spatial location of the robot reveals a clear cluster of highly-unlikely points in a particular corridor, likely to yield a region with high anomaly value, while projecting onto the angular velocity and time dimensions does not reveal any clear pattern of unlikely observations. On the other hand, Figure 4.7 shows data in which the CoBot's motion is subtly inaccurate when it turns left –i.e., its angular velocity is greater than 0; projecting onto the angular velocity dimension reveals a cluster when the angular velocity is positive (the time dimension is shown simply for ease of visualization), while projecting onto the dimensions of the robot location does not reveal any clear pattern. We thus seek an approach that can reliably project the data onto informative dimensions efficiently.

### Nominal behavior model

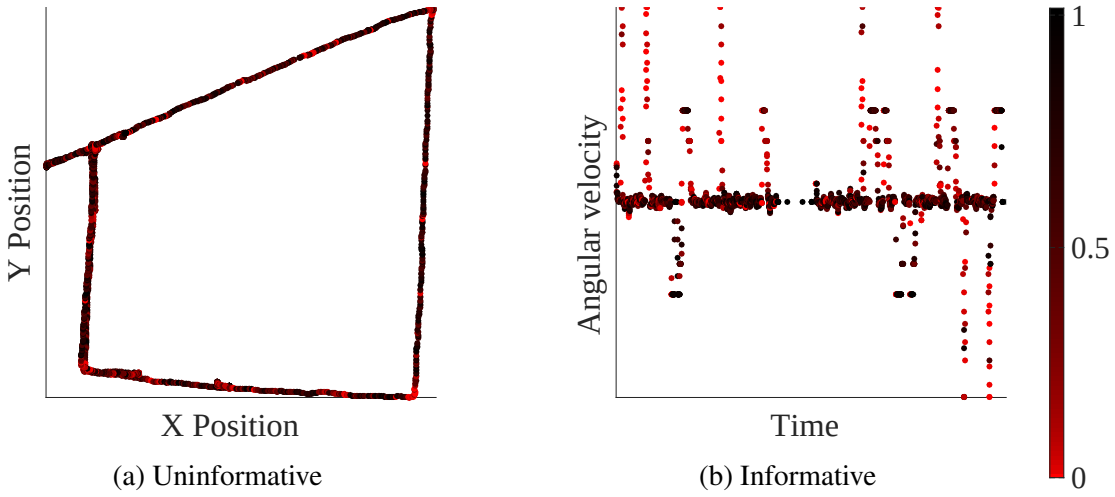
The experiments in this chapter focus on the robots' motion models, and are very similar to those in Section 3.5.2. For the nominal model, we treat the CoBot's motion as a factored MDP, and we focus on the factor that determines the CoBot's velocity $\boldsymbol{v}_{t+\Delta t}$ at time $t + \Delta t$ given its velocity $\boldsymbol{v}_t$ at time $t$ and its velocity command $\boldsymbol{u}_t$ at time $t$; $\Delta t$ is the latency of a velocity command. Our estimate of the CoBot's velocity transition function is determined by the CoBot's acceleration limit $A_{\max}$ and the covariance $\boldsymbol{\Sigma}(\boldsymbol{v}_{t+\Delta t}|\boldsymbol{v}_t, \boldsymbol{u}_t)$ of its actuator's noise obtained from nominal execution:

$$P(\boldsymbol{v}_{t+\Delta t}|\boldsymbol{v}_t, \boldsymbol{u}_t) \sim \mathcal{N}\left(\boldsymbol{\mu}(\boldsymbol{v}_{t+\Delta t}|\boldsymbol{v}_t, \boldsymbol{u}_t), \boldsymbol{\Sigma}(\boldsymbol{v}_{t+\Delta t}|\boldsymbol{v}_t, \boldsymbol{u}_t)\right), \tag{4.9}$$




(a) Informative (b) Uninformative

Figure 4.6: Data from a motion inaccuracy affecting the CoBot in a particular corridor of its domain, shown in two different projections. The likelihood of each individual observation is shown using the provided color scale.




(a) Uninformative (b) Informative

Figure 4.7: Data from a motion inaccuracy affecting the CoBot when it turns left, projected onto two different pairs of features.

where the expected velocity $\boldsymbol{\mu}(\boldsymbol{v}_{t+\Delta t}|\boldsymbol{v}_t, \boldsymbol{u}_t)$ is given by

$$\mu(\boldsymbol{v}_{t+\Delta t}|\boldsymbol{v}_t, \boldsymbol{u}_t) = \boldsymbol{v}_t + \text{bound}\left(\boldsymbol{u}_t - \boldsymbol{v}_t, A_{\max}\right). \tag{4.10}$$

Thus, as shown in Table 2.2, our nominal model $\boldsymbol{\theta}^0$ is given by

$$\boldsymbol{\theta}^0(\boldsymbol{v}_t, \boldsymbol{u}_t, \boldsymbol{v}_{t+\Delta t}) \sim \mathcal{N}\left(\boldsymbol{\mu}(\boldsymbol{v}_{t+\Delta t}|\boldsymbol{v}_t, \boldsymbol{u}_t), \boldsymbol{\Sigma}(\boldsymbol{v}_{t+\Delta t}|\boldsymbol{v}_t, \boldsymbol{u}_t)\right) \tag{4.11}$$

We note that for this model we assume that all of the stochasticity in the system comes from the CoBot's actuation, and not from its velocity measurements. We use this approximation to satisfy the fully-observable assumption throughout this thesis; Chapter 7 discusses future relaxation of this assumpption.

### High-dimensional context

The CoBot operates in an unconstrained office environment, and thus its domain is naturally high-dimensional. To vary the dimensionality of the context space in these experiments, different subsets of the robot's context space were pre-selected:

**7D context space:** Time (1D), robot estimated position (2D) and orientation (1D), linear and angular velocity commands (3D).

**15D context space:** 7D context plus robot battery voltage (1D), progress information along the current navigation graph edge (3D), depth-camera plane-extraction statistics (4D).

**30D context space:** 15D context plus depth values from 15 laser rangefinder rays, uniformly-spaced along the rangefinder's field of view.

**100D context space:** 15D context plus depth values from 85 laser rangefinder rays, uniformly-spaced along the rangefinder's field of view.

### Context-dependent model inaccuracies

Two different types of model inaccuracies were injected into the robot's motion execution:

**Corridor failure (Figure 4.6)** When moving in a particular corridor of the building, one of the robot's wheel encoders observes $0.95d$, at each timestep, where $d$ is the displacement of the wheel observed during nominal execution. Thus, the RIM encompasses non-zero velocity points in a rectangular region of physical space.

**Left turn failure (Figure 4.7)** The robot's execution is nominal except when it turns left (i.e., $\dot{\phi} > 0$), in which case each of its wheels moves only at $0.95v$ for a velocity command $v$. Since the robot usually turns only at intersections or when it needs to face a doorway, this failure mode tests the algorithm when the anomalous data is quite far apart in physical space and time, but close along the angular velocity dimension.

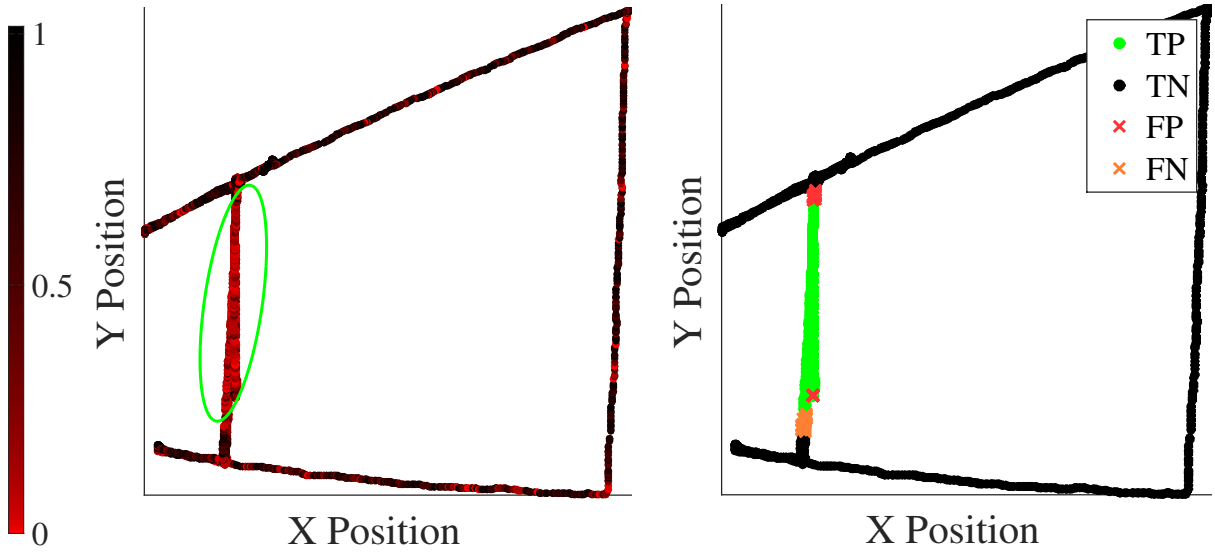
These different types of inaccuracies affect different regions of the robot's context space, thus testing the generality of our feature selection algorithm.

### Experimental procedure

For all the experiments, the CoBot was commanded, at a higher level, to navigate to various random points in the same floor of its. The chosen path, as well as lower-level behaviors like obstacle avoidance and localization are handled by pre-existing algorithms [8]. The variance of the noise in Equation 4.11 was estimated from nominal execution data captured over approximately 10 minutes of robot execution. Then, each of the testing anomalous conditions was run 10 times, each for approximately 3 minutes of execution.

### Experimental results

Figure 4.8 shows an example of a detected RIM on the Corridor Failure scenario. We note that the robot autonomously found the most informative projections for RIM-detection, as well as the RIM approximation within the projected subspace.



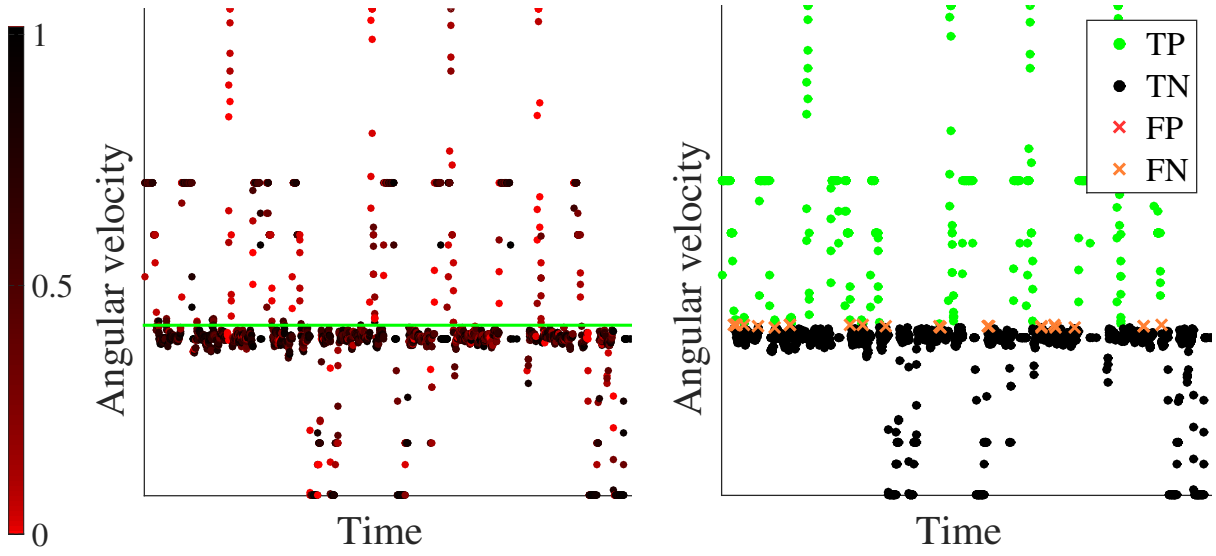

(a) Detected region

(b) Performance

Figure 4.8: Example of FS-RIM applied to the Corridor Failure. (a) An informative projection enables detection of a region containing collectively-highly-unlikely observations. (b) Detection performance.

Similarly, Figure 4.9 shows an example of FS-RIM in the Left Turn Failure scenario. In this case, FS-RIM finds the most anomalous region to lie only along the angular velocity dimension, and thus appears as the region above the horizontal green line in Figure 4.9a. Most of the False Negatives in the Left Turn Failure scenario are points with angular velocity near 0. The deviation from nominal of these points is very small, since it is proportional to the angular velocity itself; thus, these points would not increase the anomaly value of the detected RIM.

Figures 4.10 and 4.11 show the performance of FS-RIM in the Corridor Failure and Left Turn Failure scenarios respectively. Similarly to the golf-putting results of Section 4.2.3, FS-RIM enables the robot to detect RIMs in high-dimensional domains much more effectively than not using




(a) Detected region

(b) Performance

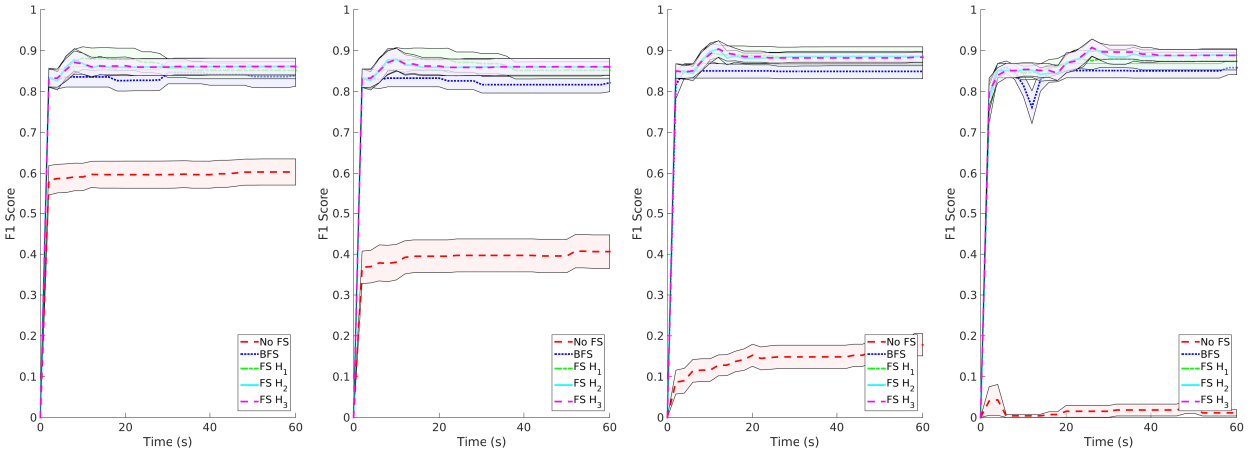
Figure 4.9: Example of FS-RIM applied to the Left Turn Failure. (a) The optimal detected region lies only along the angular velocity dimension (above the green line). (b) Detection performance.

FS-RIM. In the Corridor Failure domain, FS-RIM shows slightly better performance than BFS. In the Left Turn Failure domain their performances are equivalent; this is expected behavior, as the Left Turn Failure domain has a one-dimensional RIM (see Section 4.2.1).

In these domains, the three heuristics did not show a significant difference in performance from each other. We hypothesize that this, as well as the overall better performance of the various algorithms on these domains, is due to the higher density of data, which enables the robot to more clearly differentiate between nominal and anomalous execution. Figures 4.10d and 4.11d show a distinct shape: a quick increase in performance score, followed by a short plateau, followed by another increase and the final plateau. The first increase reflects the robot computing the anomaly value of the entire data set –i.e., the 0D projection at the root of the search tree. The first plateau happens while the robot computes the maximum anomaly region $R^+_{\{f\}}$ for each feature $f$, as required for each of the heuristics of Section 4.1.1. Finally, the next increase happens as the robot finds the right projection onto the 2D physical space in Figure 4.10d, and onto the 1D angular velocity space in Figure 4.11d.

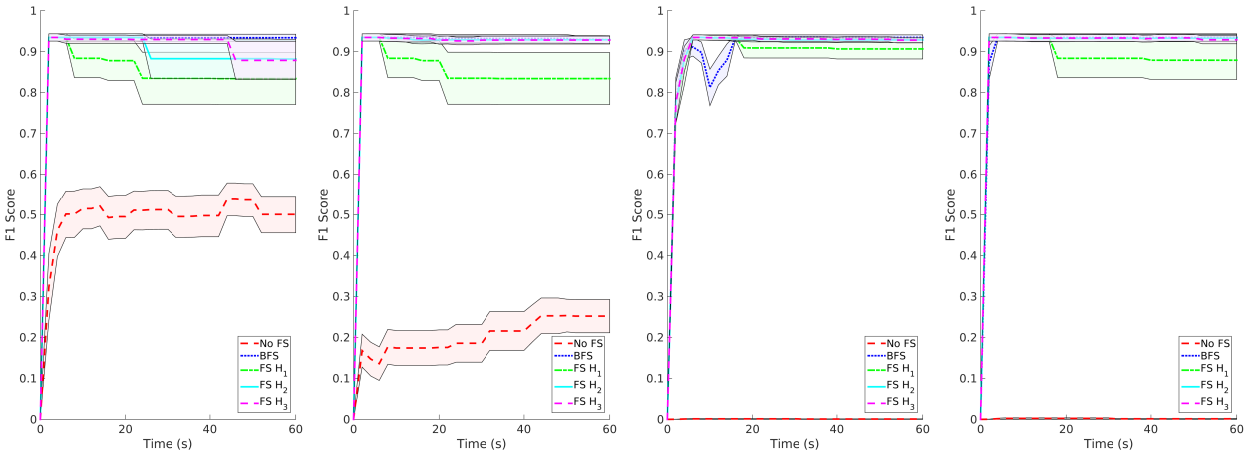
### 4.2.5 Spacecraft landing experiments

The NASA spacecraft landing simulation provides the inputs and outputs of the autonomously-landing spacecraft of Figure 1.3b on the surface of a planet. As described below, the two main challenges of this domain are: 1) The prediction model was built entirely from training data, with no analytical component; 2) the model inaccuracies in this domain were entirely unknown to the algorithm designers and trainers.




(a) 7 Dimensions (b) 15 Dimensions (c) 30 Dimensions (d) 100 Dimensions

Figure 4.10: Detection performance in the real-robot CoBot domain under a Corridor Failure. As the dimensionality of the domain increases by adding more features from execution, RIM-detection using informed Feature Selection (FS $H_i$) with various heuristics significantly outperforms not using Feature Selection (No FS), and slightly outperforms Breadth-First Search (BFS).




(a) 7 Dimensions (b) 15 Dimensions (c) 30 Dimensions (d) 100 Dimensions

Figure 4.11: Detection performance in the real-robot CoBot domain under a Left Turn Failure. As the dimensionality of the domain increases by adding more features from execution, RIM-detection using informed Feature Selection (FS $H_i$) with various heuristics significantly outperforms not using Feature Selection (No FS), but is comparable to Breadth-First Search (BFS).

### Nominal behavior model

We created a data-driven model of nominal execution of the landing spacecraft. Using a Kernel Density Estimation approach [1], we seek to predict the set of sensor readings at time $t+1$ based on the sensor readings at time $t$ and the actuator inputs at time $t$. For this thesis, we assume total observability of the environment, and thus we assume any discrepancy between predictions and observations stems from actuator noise or from model inaccuracies. We therefore model this domain as a Markov Decision Process (MDP) in which the perfect sensor readings form the state $\boldsymbol{s}$, and the actuator inputs constitute the action $\boldsymbol{a}$:

$$\boldsymbol{\theta}^0(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') = P_{\boldsymbol{\theta}^0}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}). \tag{4.12}$$

We approximate this probability as a normal distribution with the mean and variance determined by a kernel-weighed estimate:

$$\boldsymbol{\mu}(\boldsymbol{s}_{t+1}|\boldsymbol{x}_t = \chi(\boldsymbol{s}_t, \boldsymbol{a}_t)) = \frac{\sum_{i=1}^{n-1} K(\boldsymbol{x}_t, \boldsymbol{x}^i)\boldsymbol{s}^{i+1}}{\sum_{i=1}^{n-1} K(\boldsymbol{x}_t, \boldsymbol{x}^i)}, \tag{4.13}$$

where $\boldsymbol{x}^i$ are the $n$ data points observed in the training set, and $K$ is a kernel function that determines the proximity between two data points –in this case, it was a Gaussian kernel function. The variance $\boldsymbol{\Sigma}$ of the observations is obtained in a similar fashion, using a weighted sum over the squared deviation from the mean:

$$\boldsymbol{\Sigma}(\boldsymbol{s}_{t+1}|\boldsymbol{x}_t = \chi(\boldsymbol{s}_t, \boldsymbol{a}_t)) = \frac{\sum_{i=1}^{n-1} K(\boldsymbol{x}_t, \boldsymbol{x}^i)\left(\boldsymbol{s}^{i+1} - \boldsymbol{\mu}(\boldsymbol{s}_{t+1}|\boldsymbol{x}_t)\right)\left(\boldsymbol{s}^{i+1} - \boldsymbol{\mu}(\boldsymbol{s}_{t+1}|\boldsymbol{x}_t)\right)^\top}{\sum_{i=1}^{n-1} K(\boldsymbol{x}_t, \boldsymbol{x}^i)} \tag{4.14}$$

### High-dimensional context

The domain is *high-dimensional* in the space of both its inputs and outputs. The data streams from the simulation include sequences of 25-dimensional time stamped sensor data, including spacecraft velocity from various sensors, spacecraft altitude, attitude, gravity and mass sensors, and sensed position. Along with these, the data stream also includes a 4-dimensional thruster command. However, the robot has no semantic meaning attached to any of these readings, other than being able to distinguish sensor readings from commands. The monitor must find, as early as possible during execution, whether there are inconsistencies between the commands and the received sensors, and in which contexts. Thus, the context space of the domain is 29-dimensional, while the outcome space is 25-dimensional.

### Context-dependent model inaccuracies

The context-dependent model inaccuracies in this domain were injected by members of the JPL team, and were unknown to the experimenters. During various testing cases, different data streams

[1] Since our work does not focus on model-building, but rather detection of model inaccuracies, we did not seek more sophisticated modeling schemes.

were affected by context-dependent inaccuracies, causing the data to diverge significantly from its nominal behavior. The developers/trainers of the algorithm were handed various runs from 12 different types of context-dependent anomalies, with no further information about each type of anomaly.

### Experimental procedure

First, the nominal behavior model was trained with each of the 50 available nominal execution data sets. Furthermore, we used the data from these nominal runs to extract the correct value for threshold $a_{\text{thresh}}$ of Algorithm 4 in Section 4.1, as described in Section 3.2. We set this threshold value to raise an execution anomaly alarm when the confidence exceeds 95%. Figure 4.12 shows the obtained threshold, as a function of execution time. There are two periods of time at which the required threshold anomaly rises significantly: the first is when the robot decides to turn on its thrusters, which happens at somewhat different conditions even during nominal execution; the second is near the end of the simulation, in which the robot's behavior is highly variable even during nominal execution as well.

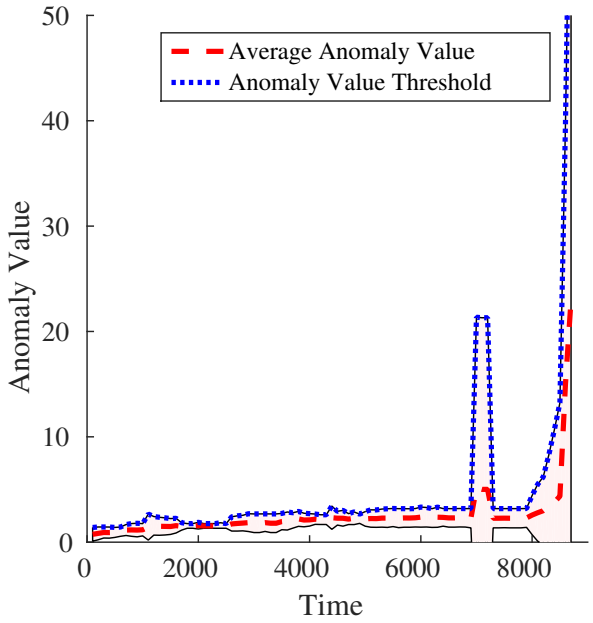



Figure 4.12: Illustration of threshold $a_{\text{thresh}}$, derived empirically from nominal spacecraft landing execution; threshold is at the 95% confidence margin.

The goal of this domain is to be able to detect anomalies early during execution. However, FS-RIM has the limitation (further discussed in Chapter 7) of not being an online algorithm. Thus, for each of the anomalous data sets, we ran our FS-RIM algorithm on increasingly large portions of the test set, increasing by 100 data points at a time, with the data sorted in chronological order. The detection algorithm finalized when an model inaccuracy was detected with 95% confidence, or when the entire data set was evaluated without finding a model inaccuracy. Since we do not possess

ground truth information about the nature of the inaccuracy, our evaluation consists entirely on the ability of FS-RIM to detect an anomaly.

**Experimental results**

Table 4.1 summarizes the results of applying FS-RIM to the spacecraft landing scenario. Of the 12 anomalies injected by a third party, 9 of them were consistently detected by our algorithm, and 2 were inconsistently detected.

| Anomaly identifier | Detection consistency | Average observations needed (total $\sim$ 8400) |
|---|---|---|
| Fault 1 | Consistent | 100 |
| Fault 2 | Consistent | 7100 |
| Fault 3 | Consistent | 7100 |
| Fault 4 | Inconsistent | – |
| Fault 5 | Consistent | 7000 |
| Fault 6 | Not detected | – |
| Fault 7 | Inconsistent | – |
| Fault 8 | Consistent | 7100 |
| Fault 9 | Consistent | 250 |
| Fault 10 | Consistent | 200 |
| Fault 11 | Consistent | 2000 |
| Fault 12 | Consistent | 2000 |

Table 4.1: Detection results from the spacecraft landing simulation domain. Of the 12 anomalies, all unknown to the robot and developers, the algorithm detected 9 every time, and 2 of them inconsistently. Only one of the anomalies went undetected consistently.

Some of the less subtle anomalies that affected the robot throughout its context space, such as faults 1, 9, and 10, were almost immediately detected by our algorithm: sensor $\mathrm{rfpi_s en}$ in fault 1 showed zero-readings randomly throughout execution; in fault 9, the sensed mass was slightly higher than predicted by the model throughout; in fault 10, the sensed altitude was consistently slightly higher than predicted by the model.

Many of the more subtle anomalies were detected with around 7000 observations, which is the time when the spacecraft turns on its thrusters. In these, the root of the anomaly is less clear, but the robot quickly detects behavior that deviates from nominal.

Future work includes further analysis of these results, with the collaboration of the third party anomaly-injecting collaborators at JPL. Given that the algorithm has produced detections, knowledge of the nature of each anomaly would enable us to evaluate the detection results against ground truth information.

## 4.3 Chapter summary

This chapter presents an algorithm for detecting RIMs in high-dimensional robot domains. The key underlying assumption of the approach, which very often holds in real robot domains, is that any RIMs are intrinsically low-dimensional, but embedded in a high-dimensional space. Thus, the solution involves a feature selection algorithm in which the robot projects its outcome observations onto low-dimensional spaces, and proceeds to detect RIMs using the algorithms of Chapter 3.

Section 4.1 presents the technical details of the contributed feature selection algorithm. The algorithm conducts a best-first search over the space of possible low-dimensional projections to find the one that contains the maximum anomaly region. This search begins in the zero-dimensional projection, and expands by adding one feature at a time to one of the explored projections. The order of this expansion depends on a heuristic anomaly value of each projection.

Section 4.2 experimentally validates the feature selection algorithm with real robot experiments. The algorithm is capable of finding model inaccuracies in various high-dimensional robot domains: the CoBot mobile robot, a NASA spacecraft landing simulation, and a high-dimensional version of the synthetic golf-robot domain. Furthermore, the algorithm vastly outperforms the low-dimensional algorithm of Chapter 3, and also outperforms an alternate feature selection approach that uses uninformed search, rather than informed search.

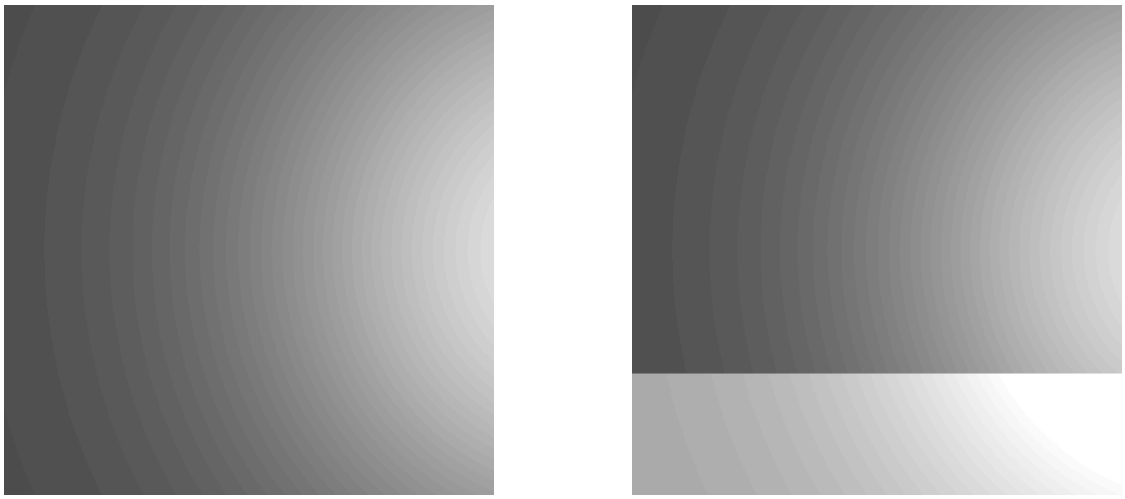
# Chapter 5

# Planning under RIM-uncertainty

This chapter discusses the problem of enabling robots to improve performance online by making plans that account for uncertainty about Regions of Inaccurate Modeling (RIMs) in their planning models. Up to this point of the thesis, the robot's actions have only been affected by the changes that our RIM-detection and correction algorithms make to the robot's models. That is, we have focused only on creating a monitor that observes the robot's execution and applies changes to its planning models. In this chapter, on the other hand, we make changes to the robot's planner, enabling it to take actions that account for RIM-uncertainty.

**Exploring better-than-expected actions.** The experiments of Chapter 3 demonstrate that robot performance can be significantly improved through passive detection and correction of RIMs: when there are action outcomes that do not meet the robot's expectations, the robot's performance is often hindered by plans that do not execute as expected. This chapter demonstrates that robot performance can be further improved by enabling robots to to find actions that lead to better-than-expected performance. Figure 5.1 shows an example of one such better-than-expected model inaccuracy in the golf-putting domain. Finding these better-than-expected inaccuracies can often only be achieved through active exploration of the robot's state-action space: the robot needs to take actions that its nominal model $\boldsymbol{\theta}^0$ does not consider optimal, but which may be optimal due to context-dependent inaccuracies in $\boldsymbol{\theta}^0$.

**Continuous Contextual Multi-Armed Bandit (CC-MAB).** In particular, this chapter focuses on the problem of online learning in the CC-MAB problem: At each time step, the robot finds itself in a given continuous-valued state $\boldsymbol{s}_t$, given which it must choose an action $\boldsymbol{a}_t$; after performing the action, the robot observes a reward $r_t$, drawn from a stochastic distribution $P^*(r|\boldsymbol{s}, \boldsymbol{a})$, which is an unknown function of the given state $\boldsymbol{s}_t$ and the chosen action $\boldsymbol{a}_t$. In particular, we use the domain of Figure 5.2, in which the soccer robot must repeatedly shoot against an opponent goalie with unknown behavior, so as to try to maximize its overall scoring reward. In this domain, the robot's state is described by the ball's position on the field at each episode; the robot must choose as an action the target on the goal line at which to shoot the ball; after the shot, the robot observes whether its shot successfully got past the goalie and into the goal –or, for a more fine-grained reward, the robot observes the distance from the goalie and the goal-posts at which the shot passed. While



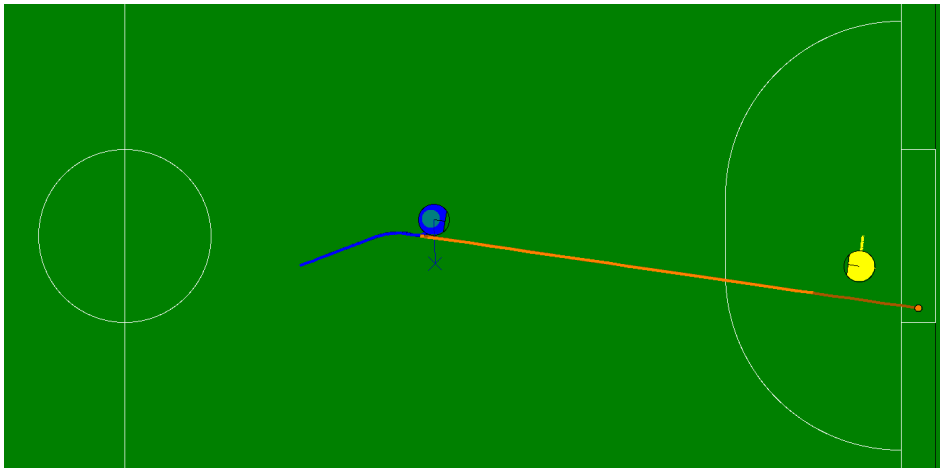
(a) Nominal performance model in context space (b) Better-than-expected model inaccuracy

Figure 5.1: Illustrative example of a better-than-expected model inaccuracy

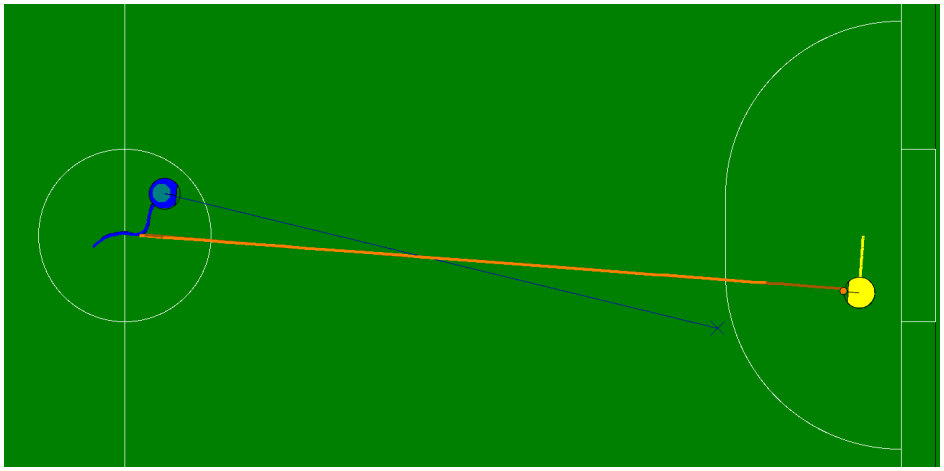
this chapter focuses on the CC-MAB, which is a specific subproblem of Reinforcement Learning (RL), many of the ideas regarding enabling robots to reason about uncertainty with respect to RIMs generalize straightforwardly to other RL problems; we will highlight these generalizations in the corresponding sections below.

**Upper Confidence Bound for CC-MAB.** Previous research has shown that the Upper Confidence Bound (UCB) algorithm is an effective solution to various types of Multi-Armed Bandit problems (e.g., [3, 85]). In particular, a state-of-the art approach to CC-MAB problems is the is the Contextual Gaussian Processes Upper Confidence Bound (CGP-UCB) algorithm [52]. This algorithm maintains an estimate of the expected reward as a function of the given state and the robot's chosen action in the form of a Gaussian Process (GP). This model for approximating the expected reward is able to provide both an estimate for the function itself as well as a measure for the uncertainty of the model at each point in the state-action space. Given these estimates, one can implement the UCB algorithm, which amounts to, at each episode, choosing the action $\boldsymbol{a}$ that maximizes a weighted sum between the expected reward $\hat{r}$ and the uncertainty $\sigma_{\hat{r}}$ about it.

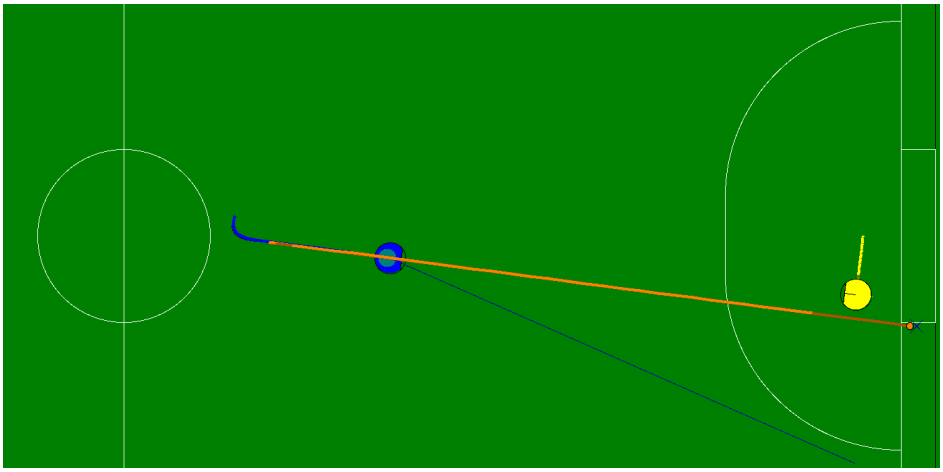
**Chapter contribution.** This chapter provides the necessary formulation to apply the UCB algorithm when the robot detects RIMs in its state-action space. To apply UCB to a domain with RIMs, we enable robots to represent their uncertainty with respect to these RIMs to obtain estimates of the expected reward $\hat{r}$ and the robot's uncertainty $\sigma_{\hat{r}}$ about that estimate. Given these estimates, the robot can apply UCB straightforwardly. Another contribution of this chapter is to combine the continuous modelling capabilities of GPs, with the discrete- or abrupt-change detection capabilities of RIM-based approaches, into a single online learning algorithm.



(a) Goal scored



(b) Shot blocked



(c) Shot out of bounds

Figure 5.2: Our robot (blue) taking shots (orange ball and trail) on the nominal behavior goalie (yellow). The robot's probability of scoring depends heavily on the shot distance from the goal. Thick blue and yellow lines show robot trajectory for the past second.

**Chapter organization.** Section 5.1 introduces some background information about the CGP-UCB algorithm. To effectively trade off exploration and exploitation, Section 5.2 extends the notion of RIMs to reason about the robot's uncertainty with respect to their spatial extent and their effect on the reward function. Finally, Section 5.3 presents empirical evidence from a physics-based simulation of the soccer-shooting domain showing that combining these two approaches yields faster learning than either approach on its own.

## 5.1 Background: Upper Confidence Bound and Contextual Gaussian Processes

In multi-armed bandits, contextual multi-armed bandits, and continuous contextual multi-armed bandits domains, the Upper Confidence Bound (UCB) algorithm has been shown to achieve no-regret online learning. In general, this approach selects, at each episode, the action that maximizes a linear combination of the action's estimated expected reward $\hat{r}$, and the robot's uncertainty $\sigma_{\hat{r}}$ about that estimate. This enables the robot to effectively trade off exploitation, by choosing actions with high estimated expected reward, and exploration, by choosing actions with high uncertainty about the estimated expected reward.

### Contextual Gaussian Process Upper Confidence Bound.

Particularly, we are interested in the continuous contextual multi-armed bandits domains –i.e., the robot is given a context $\boldsymbol{s}_t$ (the continuous state of the world, not chosen by the robot), and it chooses an action $\boldsymbol{a}_t$; a reward $r_t$ is observed immediately, and a new episode begins with the robot in a new random state. The goal of the robot is to maximize its overall reward $\sum_t r_t$.

In the case of these continuous contextual multi-armed bandits, previous research [52] has shown that a Contextual Gaussian Process UCB (CGP-UCB) algorithm achieves sub-linear regret in learning, given sufficient smoothness in the reward function. This approach estimates the expected reward, as a function of the given state $\boldsymbol{s} \in \boldsymbol{S}$ and chosen action $\boldsymbol{a} \in \boldsymbol{A}$, using Gaussian Processes (GPs) regression [79]; more generally, as in the rest of this thesis, the GP estimates the expected reward as a function of a feature vector $\boldsymbol{x}_t$ extracted from $\boldsymbol{s}_t$ and $\boldsymbol{a}_t$: $\boldsymbol{x}_t = \chi(\boldsymbol{s}_t, \boldsymbol{a}_t)$. During each episode, the algorithm chooses the action $\boldsymbol{a}_t$ that maximizes a weighted sum of the estimated expected reward function $\hat{r}(\boldsymbol{x}_t)$ and the uncertainty about this estimate $\sigma_{\hat{r}}(\boldsymbol{x}_t)$:

$$\boldsymbol{a}_t = \arg\max_{\boldsymbol{a} \in \boldsymbol{A}} \left[ \hat{r}(\chi(\boldsymbol{s}_t, \boldsymbol{a})) + \beta_t^{1/2} \sigma_{\hat{r}}(\chi(\boldsymbol{s}_t, \boldsymbol{a})) \right], \tag{5.1}$$

where $\beta_t^{1/2}$ is an appropriately-chosen coefficient to trade off exploitation and exploration [52]. While this approach has performance guarantees given a sufficiently smooth reward function, it does not for domains with discrete changes in the reward as a function of the state and action.

In our previous work [60], we have successfully applied an approach very similar to this CGP-UCB to achieve online learning of free kick policy in robot soccer –the main difference being that the robot has to select among a finite set of actions. Appendix B describes this application of GP-based online learning in detail.

## 5.2 Upper Confidence Bound for domains with RIMs

To incorporate the notion of Regions of Inaccurate Modelling (RIMs) into the UCB algorithm, we must be able to define estimates for the expected reward function $\hat{r}$, as well as the robot's uncertainty $\sigma_{\hat{r}}$ about it. As described in Section 3.4, the expected reward function can be estimated by applying the maximum likelihood shift to each RIM. However, previous work has not addressed the problem of uncertainty in RIM-modeling. Section 5.2.1 explains in detail how to describe and estimate the various types of uncertainty about RIMs; Section 5.2.2 then describes the updated computation of the expected reward estimate $\hat{r}$, while Section 5.2.3 explains the computation of the robot's uncertainty $\sigma_{\hat{r}}$ about $\hat{r}$. Section 5.2.4 brings those concepts together into the RIM-UCB algorithm, while Section 5.2.5 goes one step further and combines RIM-UCB with CGP-UCB into a RIM +GP-UCB algorithm.

### 5.2.1 Uncertainty in RIM-modelling

In RIM-detection, there are three sources of uncertainty:

**Existence** Uncertainty about the existence of a RIM in a particular point in context space

**Extent** Uncertainty about the spatial extent of each RIM in the context space

**Effect** Uncertainty about the effect of the RIM on the outcome distribution –e.g., how much is the expected reward shifted inside each RIM.

#### Existence Uncertainty

The existence uncertainty is generally captured by the anomaly value of each RIM: higher values indicate higher confidence in the existence of the RIM. As shown in Section 3.4.2, a map $\mathbb{P} : \mathbb{R} \rightarrow \mathbb{R}$ from anomaly value $\mathrm{anom}(R)$ to the probability of anomaly of a region $R$ can be obtained through monte-carlo simulations of the nominal domain:

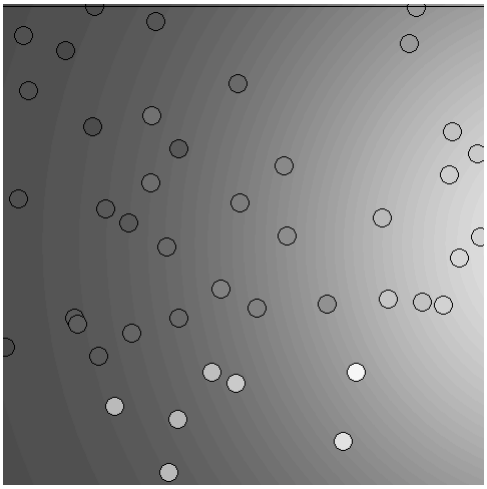
$$\mathbb{P}(\mathrm{anom}(R)) \approx P(R \text{ is anomalous}) \tag{5.2}$$

Additionally, the robot may have a prior estimate about the probability of a RIM for each point in state-action space, even before any outcome observations have been made. We capture this prior distribution as a zero-mean Gaussian Process, which is updated every time an observation is made that contradicts evidence for a RIM.

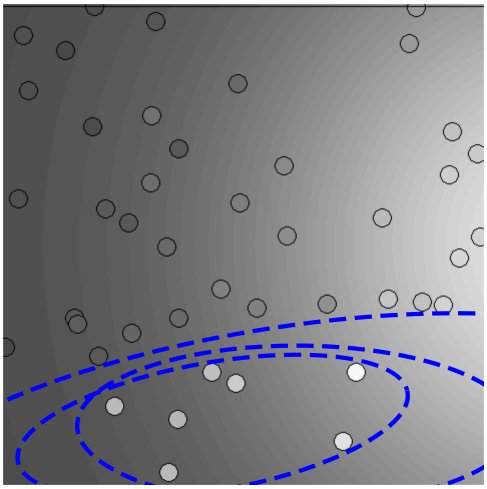
#### Extent Uncertainty

It is frequently the case, especially in domains with sparse data, that the precise extent of a RIM cannot be known exactly, even when the presence of such a RIM is known with high confidence. Figure 5.3 illustrates this problem: even though it is clear that a RIM exists around the highly-anomalous observations, there are many possible RIMs consistent with such data.

To capture the extent uncertainty, we extend the notion of a RIM by representing each RIM not as a single parametric region, but rather a set $\mathcal{R}$ of parametric regions $R^i$ that together capture



(a) Samples from the RIM of Figure 5.1b overlaid with the nominal model.



(b) Examples of various parametric regions consistent with the execution data.

Figure 5.3: Illustration of the extent uncertainty problem: although it is clear from the data that a RIM exists, there are many parametric regions consistent with the data.

an estimate of the distribution over possible extents of the RIM. In particular, in the case of a non-subtle RIM like that of Figure 5.3a, we want the set $\mathcal{R}$ to contain at least one of the *most specific consistent hypotheses* and at least one of the *most general consistent hypotheses* [36][1], as illustrated in Figure 5.4.

To define the set $\mathcal{R}$, we first define a modified anomaly cost function $\text{anom}^+(R, \lambda)$ that trades off between a region's anomaly value and their size, according to a parameter $\lambda$:

$$\text{anom}^+(R, \lambda) = \text{anom}(R) + \lambda V(R), \tag{5.3}$$

where $V(R)$ represents the volume of region $R$. Then, each RIM is thus represented by a set $\mathcal{R}$ of regions $R^i$ that maximize cost functions with different rewards for smaller and larger region sizes:

$$\Lambda \equiv \{\lambda_{\min}, \lambda_{\min} + \Delta\lambda, \ldots, \lambda_{\max} - \Delta\lambda, \lambda_{\max}\} \tag{5.4}$$

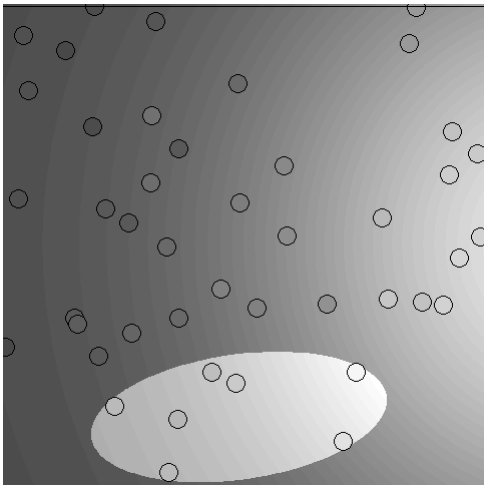
$$\mathcal{R} \equiv \left\{ \arg\max_R \left[\text{anom}^+(R, \lambda_i)\right] \,|\lambda_i \in \Lambda \right\} \tag{5.5}$$

Figure 5.3b shows an illustration of what such a set of regions may look like; Section 5.3 contains actual depictions from a robot domain.

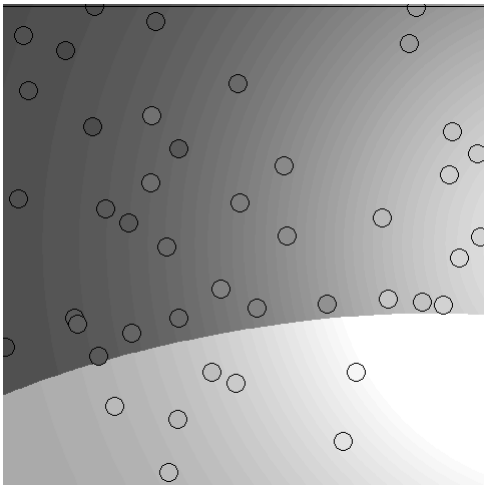
This set of regions $\mathcal{R}$, when appropriately weighed, approximates the distribution over possible spatial extents of the RIM. To achieve this approximation, each parametric region is weighed proportionally to its relative probability of being a true anomaly. Then, the probability that a point $\boldsymbol{x}$ is in RIM $\mathcal{R}$ is:

$$P(\boldsymbol{x} \in \mathcal{R}) \approx \frac{\sum_{R^i \in \mathcal{R}} \mathbb{P}(\text{anom}(R^i) \mathbb{1}(\boldsymbol{x} \in R^i)}{\sum_{R^i \in \mathcal{R}} \mathbb{P}(\text{anom}(R^i))}. \tag{5.6}$$

[1] In the case of ellipsoids, there is no single most specific or most general consistent hypothesis.



(a) Most specific consistent RIM-hypothesis



(b) Most general consistent RIM-hypothesis

Figure 5.4: Illustration of one each of the most specific and most general RIM-hypotheses consistent with the observed execution data.

Expressing each RIM as a distribution of parametric regions increases the expression power of the approach: whereas previous work on RIM-detection treats each observation as either belonging to a RIM or not, this approach enables the robot to reason about points in state-action space that have some probability between 0 and 1 of belonging to the RIM.

**Effect Uncertainty**

The effect uncertainty describes the uncertainty about how the behavior of the world is different in a RIM than what the nominal model $\boldsymbol{\theta}^0$ predicts. This thesis focuses on effects that shift the mean of the reward function by a constant vector $\boldsymbol{\delta}_{\max}$ (see Section 3.1); thus, the effect uncertainty of the effect of an individual parametric region $R$ is defined by the estimated standard error of the observed shifts $\boldsymbol{\delta}_t \equiv (\boldsymbol{z}_t - \boldsymbol{\mu}_t(\boldsymbol{\theta}^0))$ from the most likely shift $\boldsymbol{\delta}_{\max}$:

$$\sigma^2_{\hat{r}}(R|\boldsymbol{Z}, \boldsymbol{\theta}^0) \approx \frac{1}{n}\left(\frac{1}{n-1}\sum_{\boldsymbol{x}_t \in R}(\boldsymbol{\delta}_t - \boldsymbol{\delta}_{\max})^2\right) \tag{5.7}$$

To extend this formulation to a distribution $\mathcal{R}$, each parametric region is weighed according to its probability of containing the query context $\boldsymbol{x}$

$$\sigma^2_{\hat{r}}(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}, \boldsymbol{\theta}^0) \approx \frac{\sum_{R^i \in \mathcal{R}} \mathbb{P}(\text{anom}(R^i))\mathbb{1}(\boldsymbol{x} \in R^i)\sigma^2_{\hat{r}}(R^i|\boldsymbol{Z}, \boldsymbol{\theta}^0)}{\sum_{R^i \in \mathcal{R}} \mathbb{P}(\text{anom}(R^i))\mathbb{1}(\boldsymbol{x} \in R^i)} \tag{5.8}$$

### 5.2.2 Estimation of the expected reward

The UCB algorithm requires the robot to be able to estimate, for each point $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$, the expected reward, as well as the robot's uncertainty about that estimate.

The expected reward is a mixture model between the original expected reward function $\hat{r}_0$, potentially augmented with the GP corrections, and the estimated reward $\hat{r}_1$ determined by the RIM $\mathcal{R}$ (this extends straightforwardly to multiple RIMs). We estimate the probability distribution of the reward function, given a context $\boldsymbol{x} = \chi(\boldsymbol{s}, \boldsymbol{a})$, as:

$$\hat{r}(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}) = \hat{r}_0(\boldsymbol{x}|\boldsymbol{Z})(1 - P(\boldsymbol{x} \in \mathcal{R})) + \hat{r}_1(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z})P(\boldsymbol{x} \in \mathcal{R}), \tag{5.9}$$

where $P(\boldsymbol{x} \in \mathcal{R})$ is the probability of the chosen point being in $\mathcal{R}$, as defined in Equation 5.6. We can rewrite these probabilities as $w_0$ and $w_1$ to highlight that this distribution is a mixture-model:

$$\hat{r}(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}) = w_0\hat{r}_0(\boldsymbol{x}|\boldsymbol{Z}) + w_1\hat{r}_1(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}) \tag{5.10}$$

### 5.2.3 Variance of expected reward

The variance of a one-dimensional mixture distribution with weights $w_i$, means $\mu_i$ and variances $\sigma_i^2$ is given by

$$\sigma^2 = \sum_i w_i((\mu_i - \mu)^2 + \sigma_i^2), \tag{5.11}$$

where $\mu$ is the mean of the mixture distribution. Thus, in our mixture distribution specifically, the variance is given by

$$\begin{aligned}\sigma^2_{\hat{r}}(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}) =& w_0((\hat{r}_0(\boldsymbol{x}|\boldsymbol{Z}) - \hat{r}(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}))^2 + \sigma^2_{\hat{r}_0}(\boldsymbol{x}|\boldsymbol{Z}))+ \\ & w_1((\hat{r}_1(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}) - \hat{r}(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}))^2 + \sigma^2_{\hat{r}_1}(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z})),\end{aligned} \tag{5.12}$$

where $w_0, w_1, \hat{r}_0, \hat{r}_1, \hat{r}$ are defined as above, $\sigma^2_{\hat{r}_0}$ is the uncertainty of the original model –e.g., the GP variance– and $\sigma^2_{\hat{r}_1}$ is the estimated standard error of Equation 5.7.

### 5.2.4 RIM-UCB algorithm

Having a notion of the expected reward function as well as the robot's uncertainty about it enables us to apply the UCB algorithm straightforwardly, as Algorithm 5 shows. The algorithm first finds the distribution of possible RIMs. Then, it computes the expected reward and uncertainty about it. In practice, line 3 may not have a closed-form solution. In our experiments, we discretize the action space to search for the one with maximum value.

### 5.2.5 RIM +GP-UCB algorithm

While we have formulated the UCB algorithm in the context of RIM-detection, here we seek to combine the benefits of RIM-UCB –i.e., discrete behavior transitions– with those of CGP-UCBP –i.e., continuous model corrections.

To combine these algorithms into a RIM +GP-UCB algorithm, we create a GP that stores the residuals $\Delta\boldsymbol{Z}_{\boldsymbol{\theta}^+}$ of each outcome observation with respect to the corrected model $\boldsymbol{\theta}^+$ of our $RIM$-detector. Algorithm 6 shows the procedure of detecting RIMs and creating the residual GP $G$.

**Algorithm 5** RIM-UCB algorithm for online learning in the presence of RIMs in Continuous Contextual Multi-Armed Bandit problems.
*Input:* world state $\boldsymbol{s}_t$, nominal model $\boldsymbol{\theta}^0$, set of contextual observations $\boldsymbol{Z}$.
*Output:* chosen action $\boldsymbol{a}_t$.

**function** RIM-UCB($\boldsymbol{s}_t, \boldsymbol{\theta}^0, \boldsymbol{Z}$)
  $\mathcal{R} \leftarrow \text{FindRIMs}(\boldsymbol{Z}, \boldsymbol{\theta}^0)$ ▷ Find RIM distribution $\mathcal{R}$ (Chapter 3 + Eq. 5.5)
  $\boldsymbol{a}_t = \arg\max_{\boldsymbol{a} \in \boldsymbol{A}} \left[ \hat{r}(\chi(\boldsymbol{s}_t, \boldsymbol{a}) | \mathcal{R}, \boldsymbol{Z}) + \beta_t^{1/2} \sigma_{\hat{r}}(\chi(\boldsymbol{s}_t, \boldsymbol{a}) | \mathcal{R}, \boldsymbol{Z}) \right]$ ▷ UCB + Eq. 5.9, 5.12
  **return** $\boldsymbol{a}_t$
**end function**

Given this procedure and the resulting RIM-distribution $\mathcal{R}$ and residual GP $G$, the robot procedes to choose the optimal action very similarly to Algorithm 5. However, the mean and variance of the estimated distribution use the residual mean $\hat{r}^G(\boldsymbol{x})$ and uncertainty $(\sigma_{\hat{r}}^G(\boldsymbol{x}))^2$ from the GP as well as those from the RIMs:

$$\begin{aligned}
\hat{r}_0(\boldsymbol{x}|\boldsymbol{Z}, G) \equiv& \hat{r}_0(\boldsymbol{x}|\boldsymbol{Z}) + \hat{r}^G(\boldsymbol{x}) \\
\hat{r}_1(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}, G) \equiv& \hat{r}_1(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}) + \hat{r}^G(\boldsymbol{x}) \\
\hat{r}(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}, G) =& \hat{r}_0(\boldsymbol{x}|\boldsymbol{Z}, G)(1 - P(\boldsymbol{x} \in \mathcal{R})) + \hat{r}_1(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}, G) P(\boldsymbol{x} \in \mathcal{R}),
\end{aligned} \tag{5.13}$$

Similarly, we obtain the combined variance:

$$\begin{aligned}
\sigma_{\hat{r}_0}^2(\boldsymbol{x}|\boldsymbol{Z}, G) \equiv& (\sigma_{\hat{r}}^G(\boldsymbol{x}))^2 \\
\sigma_{\hat{r}_1}^2(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}, G) \equiv& \sigma_{\hat{r}_1}^2(\mathcal{R}, \boldsymbol{x}|\boldsymbol{Z}) + (\sigma_{\hat{r}}^G(\boldsymbol{x}))^2 \\
\sigma_{\hat{r}}^2(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}, G) =& w_0((\hat{r}_0(\boldsymbol{x}|\boldsymbol{Z}, G) - \hat{r}(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}, G))^2 + \sigma_{\hat{r}_0}^2(\boldsymbol{x}|\boldsymbol{Z}, G)) + \\
& w_1((\hat{r}_1(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}, G) - \hat{r}(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}, G))^2 + \sigma_{\hat{r}_1}^2(\boldsymbol{x}|\mathcal{R}, \boldsymbol{Z}, G)),
\end{aligned} \tag{5.14}$$

**Algorithm 6** Augmented RIM-detection function that also returns a residual Gaussian Process.
*Input:* Contextual observations $\boldsymbol{Z}$, model $\boldsymbol{\theta}^0$ of nominal execution.
*Output:* RIM-distribution $\mathcal{R}$, residual Gaussian Process $G$.

**function** FindRIMs $GP(\boldsymbol{Z}, \boldsymbol{\theta}^0)$
  $\mathcal{R} \leftarrow \text{FindRIMs}(\boldsymbol{Z}, \boldsymbol{\theta}^0)$ ▷ Find RIM distribution $\mathcal{R}$ (Chapter 3 + Eq. 5.5)
  $\boldsymbol{\theta}^+ \leftarrow \text{UpdateModel}(\boldsymbol{\theta}^0, \mathcal{R})$ ▷ Correct model
  $\Delta \boldsymbol{Z}_{\boldsymbol{\theta}^+} \leftarrow \left\{ \left( \boldsymbol{x}_t, \boldsymbol{z}_t - E[\boldsymbol{\theta}^+(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{z})] \right) | (\boldsymbol{x}_t, \boldsymbol{z}_t) \in \boldsymbol{Z} \right\}$ ▷ Compute residuals
  $G \leftarrow \text{BuildGP}\left(\Delta \boldsymbol{Z}_{\boldsymbol{\theta}^+}\right)$ ▷ Create residual GP
  **return** $(\mathcal{R}, G)$
**end function**

Thus, the Gaussian Process models the deviation of the reward function from the corrected model $\boldsymbol{\theta}^+$, rather than the deviation from the nominal model $\boldsymbol{\theta}^0$. We evaluate this combined approach in the experiments of Section 5.3.

**Algorithm 7** RIM +GP-UCB algorithm. Combines RIM-UCB (Algorithm 5) with CGP-UCB [52].
*Input:* world state $\boldsymbol{s}_t$, nominal model $\boldsymbol{\theta}^0$, set of contextual observations $\boldsymbol{Z}$.
*Output:* chosen action $\boldsymbol{a}_t$.

**function** RIM-UCB($\boldsymbol{s}_t, \boldsymbol{\theta}^0, \boldsymbol{Z}$)
  $(\mathcal{R}, G) \leftarrow \text{FindRIMs}\, GP(\boldsymbol{Z}, \boldsymbol{\theta}^0)$ ▷ Find RIM distribution $\mathcal{R}$ (Eq. 5.5)
  $\boldsymbol{a}_t = \arg\max_{\boldsymbol{a} \in \boldsymbol{A}} \left[ \hat{r}(\chi(\boldsymbol{s}_t, \boldsymbol{a}) | \mathcal{R}, \boldsymbol{Z}, G) + \beta_t^{1/2} \sigma_{\hat{r}}(\chi(\boldsymbol{s}_t, \boldsymbol{a}) | \mathcal{R}, \boldsymbol{Z}, G) \right]$ ▷ Eq. 5.13, 5.14
  **return** $\boldsymbol{a}_t$
**end function**

## 5.3 Empirical Evaluation

This section describes the empirical evaluation of our online learning algorithm using the RIM-UCB and RIM +GP-UCB algorithms on a Continuous Contextual Multi-Armed Bandit problem. Section 5.3.1 describes the robot soccer domain on which we apply our algorithm; Section 5.3.4 describes the experimental setup, and Section 5.3.5 describes the experimental results.

We evaluate the contributed algorithms on a two-dimensional robot soccer domain. Applying the RIM-UCB algorithm to high-dimensional robot domains is straightforward using the feature seclection approach of Chapter 4. Applying the RIM +GP-UCB algorithm to high-dimensional domains is less straightforward, since the performance of Gaussian Processes tends to diminish significantly with the dimensionality of the domain. However, the problems of online learning and Bayesian Optimization in higher-dimensions are actively being explored in the literature, for example, by assuming that the high-dimensional function to be learned can be decomposed into low-dimensional additive components [44].

### 5.3.1 Online learning domain: Scoring on an opponent goalie

To evaluate our online learning algorithm, we address a sub-problem of the robot soccer domain. A single offense robot must repeatedly shoot on the opponent's goal, with the intention of scoring as many goals as possible, despite the presence of an opponent goalie.

#### State-action space

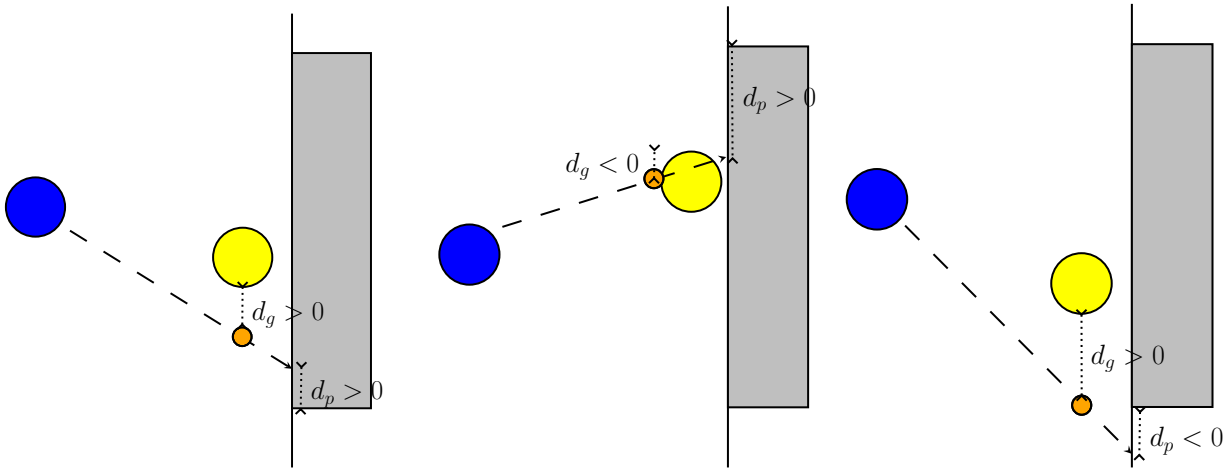
The initial location of the ball on the field is always along the center line that joins the opponents' goal to our goal, and its position $\boldsymbol{s}$ along that line is randomly given to the robot at the beginning of each episode. The robot cannot dribble or pass the ball; it must shoot the ball directly from location $\boldsymbol{s}$ to the opponent's goal. The robot always shoots at the maximum allowed speed of $8m/s$, but it can choose the point $\boldsymbol{a}$ on the goal line at which to aim its shot. Thus, the state-action space of this problem is two-dimensional and therefore easily visualizable.

### Reward

For this task, we do not use a simple 0-1 reward depending on whether the robot scores a goal. To facilitate faster learning, we instead use a continuous reward that is proportional to the distance at which the shot passes from the goalie and the goal posts. That is, given that the ball passes by the goalie at a distance $d_g$ and it crosses the goal line at a distance $d_p$ from the closest post, the reward is given by:

$$r = \min(d_g, d_p). \tag{5.15}$$

This reward function is illustrated in Figure 5.5. If the ball enters the goal (Figure 5.5a), the reward is greater than 0. If the ball is blocked by the goalie, the reward is negative (Figure 5.5b). Similarly, if the ball leaves the field outside of the goal, the reward is negative (Figure 5.5c). The distribution of the reward is intrinsically noisy, due to the noise in actuation of the shooting robot: its heading angle at the time of the shot has normal noise, whose variance is estimated from training data.




(a) Positive reward (goal) (b) Negative reward (goalie block) (c) Negative reward (goal miss)

Figure 5.5: Reward function illustration for the soccer shooting domain. The minimum signed distance between the ball (orange small circle) and the goalie (large yellow circle) $d_g$ or between the ball and the closest goal post (rectangle edge) $d_p$ determine the reward.

### Nominal outcome model

Based on a prediction of the goalie's behavior, and knowledge of the noise in actuation while shooting, the offense robot can create a stochastic nominal model of the expected reward for each possible shot. In accordance with Table 2.2, the nominal model of our robot is given by:

$$\boldsymbol{\theta}^0(\boldsymbol{s}, \boldsymbol{a}, r) = P_{\boldsymbol{\theta}^0}(r|\boldsymbol{s}, \boldsymbol{a}) \tag{5.16}$$

We approximate this distribution as a normal distribution with mean $\mu_0(r|\boldsymbol{s}, \boldsymbol{a})$ and variance $\sigma_0^2(r|\boldsymbol{s}, \boldsymbol{a})$.

**Nominal model mean.** The mean of the distribution is given by the minimum between the distance $d_p(\boldsymbol{a})$ between the goal post and the intended target $\boldsymbol{a}$, and the distance $d_g(\boldsymbol{a})$ that the ball is projected to pass from the goalie, if shot at its intended target $\boldsymbol{a}$:

$$\mu_0(r|\boldsymbol{s},\boldsymbol{a}) = \min(d_p(\boldsymbol{a}), d_g(\boldsymbol{a})). \tag{5.17}$$

Computing $d_p(\boldsymbol{a})$ is trivial, since the goal post does not move. Computing $d_g(\boldsymbol{a})$ requires modeling the opponent's behavior, and depends on the initial distance $d_g^0(\boldsymbol{a})$ from the goalie to point $\boldsymbol{a}$, and the maximum distance $d_g^{\max}(t_{bg}(\boldsymbol{a}))$ that the goalie can travel in the time $t_{bg}(\boldsymbol{a})$ it takes the ball to reach the goalie's path after being shot toward $\boldsymbol{a}$:

$$d_g(\boldsymbol{a}) = \max\left(0, d_g^0(\boldsymbol{a}) - d_g^{\max}(t_{bg}(\boldsymbol{a}))\right) - R_\rho - R_b, \tag{5.18}$$

where $R_\rho$ is the goalie's radius and $R_b$ is the ball's radius.

To compute distance $d_g^{\max}(t_{bg}(\boldsymbol{a}))$, our robot assumes that the opponent goalie will use its maximum acceleration $A_{\max}$ in a parallel direction to the goal line to attempt to intercept incomping shots. First, the robot computes the time $t_{bg}(\boldsymbol{a})$ that the ball will take between the beginning of the shot and when it passes by the goalie; this time can be derived from the distance $d_{bg}(\boldsymbol{a})$ that the ball must travel, the sliding and rolling friction coefficients of the surface on which it travels, and the equations of motion of a sliding and rolling ball [41]. The maximum travel distance of the goalie in time $t_{bg}(\boldsymbol{a})$ is given by:

$$d_g^{\max}(t_{bg}(\boldsymbol{a})) = \frac{1}{2} A_{\max}(t_{bg}(\boldsymbol{a}) - t_{\text{lat}})^2, \tag{5.19}$$

where $t_{\text{lat}}$ is the estimated latency of the opponent goalie's reaction to a shot. This assumes the goalie starts with 0 speed, and it does not reach its maximum speed. Both of these are reasonable assumptions in our robots, but the model can be straightforwardly relaxed.

**Nominal model variance.** To estimate the variance of the reward distribution, we note that the *angle* at which the robot shoots is approximately normally distributed around the angle at which it intends to shoot, with constant variance $\sigma_\alpha^2$, which we estimate from training data. Given that the angle of the shot is normally distributed with constant variance $\sigma_\alpha^2$, the distance from the intended target $\boldsymbol{a}$ at which the ball crosses the goal line is also approximately normally distributed, but with a variance that depends on the distance between the shot source $\boldsymbol{s}$ and the target $\boldsymbol{a}$:

$$\sigma_0(r|\boldsymbol{s},\boldsymbol{a}) \approx \tan(\sigma_\alpha) d(\boldsymbol{s},\boldsymbol{a}) \tag{5.20}$$

Thus, the nominal model of the robot is its reward distribution:

$$P_{\boldsymbol{\theta}^0}(r|\boldsymbol{s},\boldsymbol{a}) \sim \mathcal{N}\left(\mu_0(r|\boldsymbol{s},\boldsymbol{a}), \sigma_0(r|\boldsymbol{s},\boldsymbol{a})\right) \tag{5.21}$$

### 5.3.2 Opponent model RIMs: Goalie vulnerabilities

To evaluate the ability of our FS-RIM algorithm to find opponent weaknesses, we test it on goalies with different vulnerabilities. Figure 5.2 shows the nominal behavior of the goalie, in which it applies maximum acceleration to attempt to block each shot. We evaluate our algorithms against vulnerable goalies that deviate from this nominal behavior in different ways:

**Region goalie (Figure 5.6)** In a particular region (marked with black lines) of the state-action space of the shooting robot, the shots yield significantly more reward than expected. This condition tests a clear discontinuity in the reward distribution of the domain.

**Obstructed goalie (Figure 5.7):** There is an obstruction on the field that does not allow the goalie to see the ball when the ball is behind the obstruction. As a result, the goalie remains in its default position while it cannot perceive the ball, instead of blocking the incoming shot. Thus, for some region of state-action space, the goalie only perceives the incoming shot once it is too late to block it.

**Overshoot goalie (Figure 5.8):** An opponent goalie that the CMDragons encountered in 2015 had the following flawed policy: During each timestep of an incoming shot, apply maximum acceleration toward the path of the ball. This policy caused the goalie to sometimes overshoot the ball trajectory since it would not start decelerating until it was aligned with the ball trajectory. We call this weakness the *overshoot goalie*.

Each of these opponent weaknesses affects the behavior of the world in different sets of contexts, unknown to our robot except through its execution outcome observations.

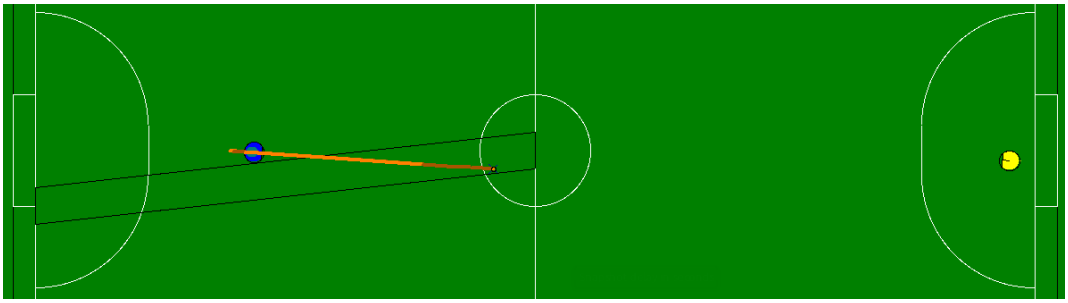


Figure 5.6: Region-Goalie behavior: When the state-action $(\boldsymbol{s}, \boldsymbol{a})$ of the robot is within the black parallelogram –where the $x$ axis shows the ball initial position $\boldsymbol{s}$ along the mid-line and the $y$ axis shows the $y$ value of the chosen target $\boldsymbol{a}$ on the goal line– the goalie concedes significantly higher reward than expected.

### 5.3.3 Planning policies

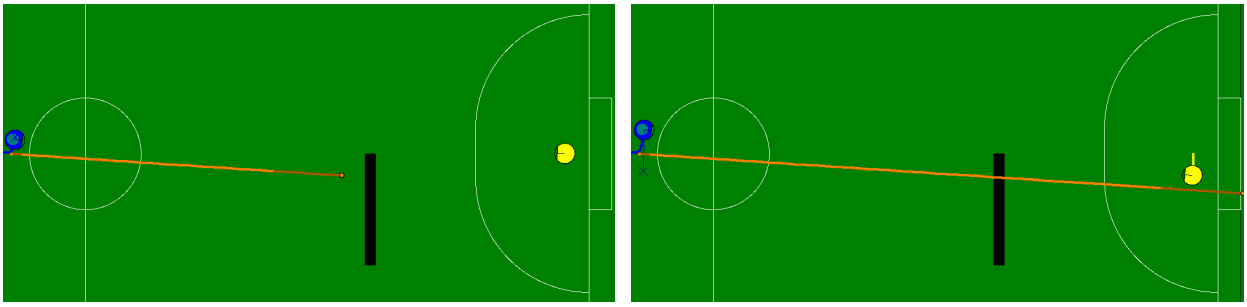
We wish to evaluate different planning policies to compare their performance as a function of time:

**Upper Confidence Bound using Gaussian Processes (GP-UCB):** Our robot chooses its action according to the CGP-UCB algorithm, as described in Section 5.1.

**Upper Confidence Bound using RIMs (RIM-UCB):** Our robot chooses its action according to the RIM-UCB algorithm of Section 5.2.4.

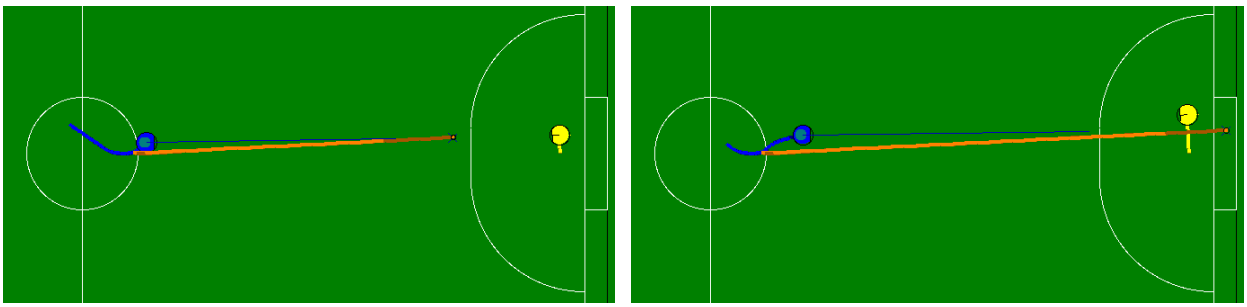
**Upper Confidence Bound using RIMs and Gaussian Processes (RIM +GP-UCB):** Our robot chooses its action using a combination of RIM and GP modeling, as described in Section 5.2.5.

For all of these policies, if there are multiple actions that maximize the value function, the robot chooses one at random.



(a) Goalie obstructed by obstacle (b) Goalie no longer obstructed, fails to move in time

Figure 5.7: Obstructed Goalie behavior: The goalie cannot perceive the ball when the black obstacle lies between itself and the ball. Thus, it is unable to block some otherwise-blockable shots.



(a) Goalie moves in the right direction to block (b) Goalie fails to decelerate in time, misses the ball

Figure 5.8: Overshoot Goalie behavior: The goalie always applies maximum acceleration toward the ball path, and thus sometimes fails to decelerate in time.

### 5.3.4 Experimental setup

We evaluate the online learning algorithm on a physX-based simulation of the CMDragons. Each experimental run is a sequence of episodes that proceed as follows:

1. The ball is automatically placed in a random location $\boldsymbol{s}_t$ along the center-line joining the center of the two goals, and our robot is placed in a random location on the field.
2. The opponent goalie takes its preferred position given $\boldsymbol{s}_t$.
3. Our robot is allowed to take a shot $\boldsymbol{a}_t$ on the opponent goalie.
4. Once the shot is complete, either because it has crossed the goal line or because it has changed direction after hitting the goalie or the goal post, the robot observes its reward $r_t$ perfectly, and a new episode begins.
5. If our robot does not take a shot within 10 seconds, a new episode begins.

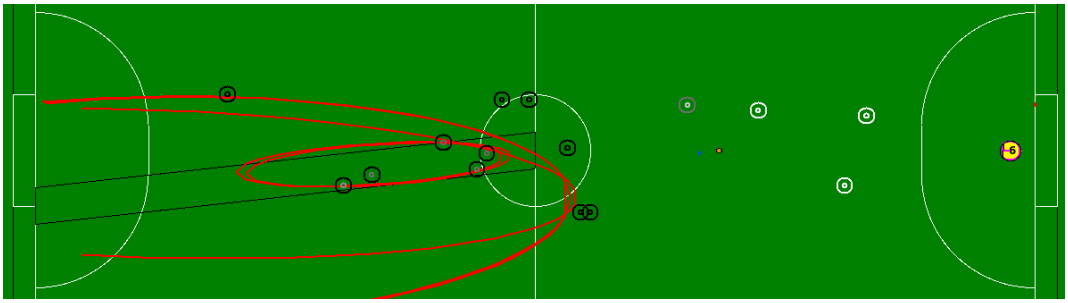
Throughout each experimental run, we record the state $\boldsymbol{s}_t$, action $\boldsymbol{a}_t$ and reward $r_t$ in each episode. We conduct 10 runs for each of the goalie conditions of Section 5.3.2, for each of the planning policies of Section 5.3.3.

### 5.3.5 Experimental results

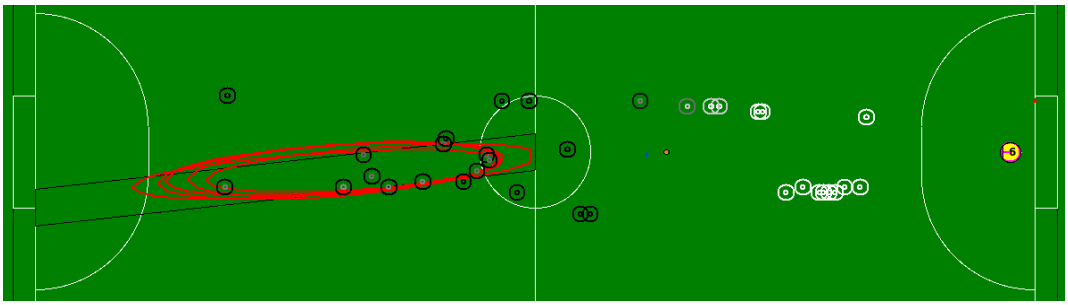
This section presents the results of running the different exploration policies on the different vulnerable goalies. We draw two general conclusions: 1) The performance of each policy significantly depends on the smoothness properties of the reward function in each domain, and 2) In general, combining our RIM-based approach with the GP-based approach provides robust performance across domains with different smoothness properties.

**Region Goalie.** Figure 5.9 shows an example of our RIM-UCB algorithm running on the Region Goalie domain. Figure 5.10a shows the reward obtained by the different policies on the Region Goalie. This domain has a very discontinuous transition in the performance of the robot over its state-action space, which is reflected in a significant difference in performance between the algorithms: The RIM-UCB and RIM +GP-UCB algorithms outperform the GP-UCB algorithm. Interestingly, RIM-UCB learns significantly faster than RIM +GP-UCB. We hypothesize this is because RIM-UCB only corrects the model when it has higher certainty of a model inaccuracy, and thus is less suceptible to make incorrect corrections based on noise early on.

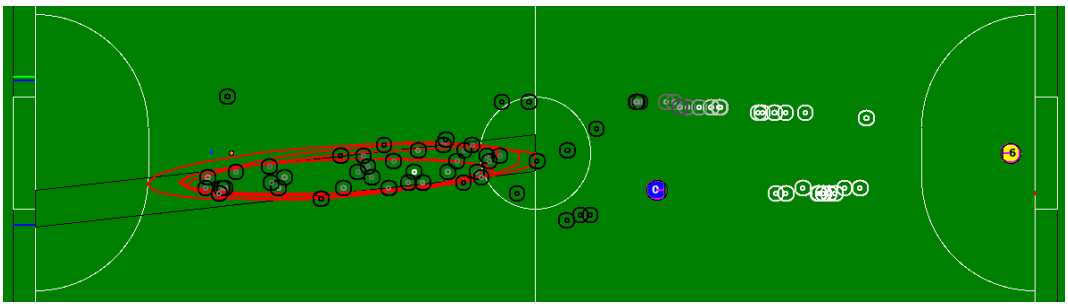
**Obstructed Goalie.** Figure 5.11 shows an example of our RIM-UCB algorithm running on the Obstructed Goalie domain. Figure 5.10b shows the results of the different learning policies on the Obstructed Goalie. This domain has an abrupt, but not discontinuous transition in the performance of the robot over its state-action space: when the robot takes a shot such that the ball is behind the obstruction, its chances of scoring are significantly higher than shooting just a bit differently, such that the ball is not obstructed from the goalie. In this domain, all of the policies perform similarly, within the margin of error of each other.



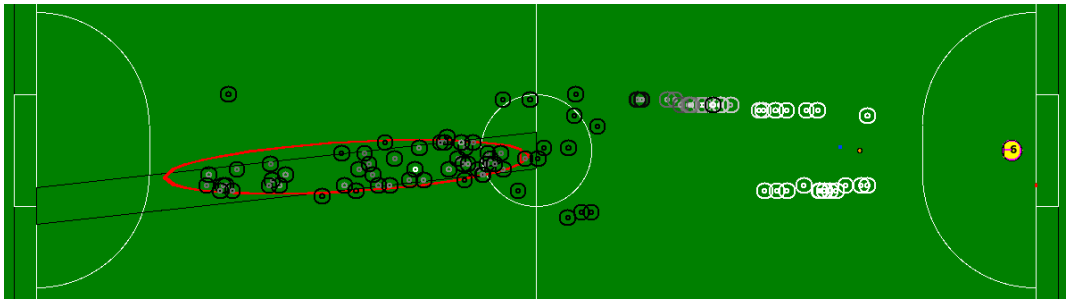
(a) 14 outcome observations



(b) 24 outcome observations

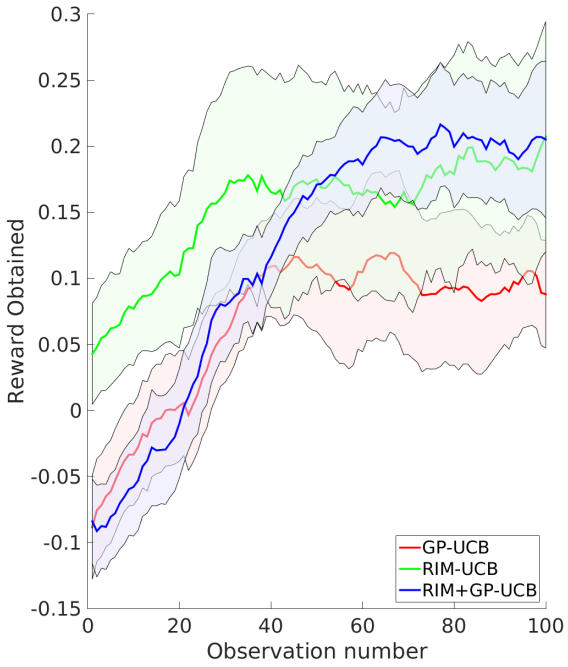


(c) 44 outcome observations

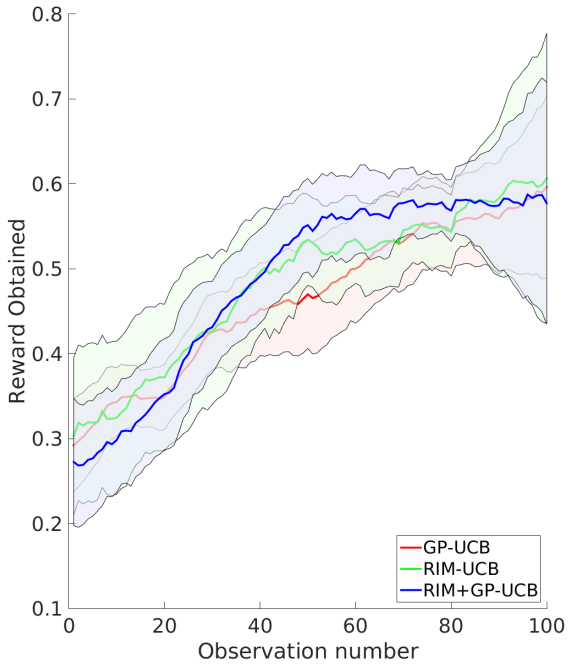


(d) 100 outcome observations

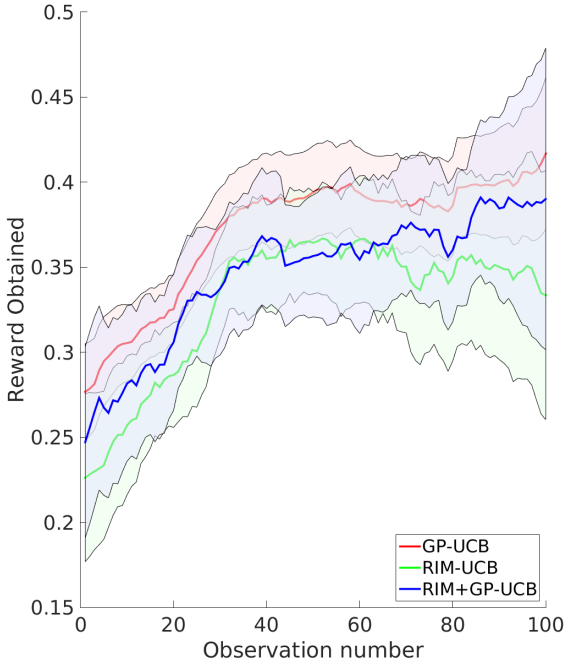
Figure 5.9: Example of the most anomalous region distribution (red ellipses) found during execution in the Region Goalie domain. Each pair of concentric circles shows an outcome observation: position shows the starting ball position $\boldsymbol{s}$ ($x-$axis) and chosen action $\boldsymbol{a}$ ($y-$axis); the outer and inner circles show the expected reward $\hat{r}_0$ from the nominal model and the observed reward $r_t$ respectively (darker is lower).




(a) Region-Goalie

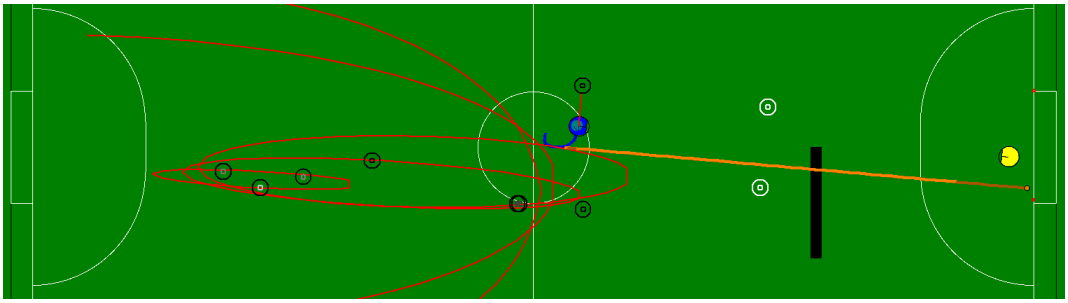



(b) Obstructed Goalie

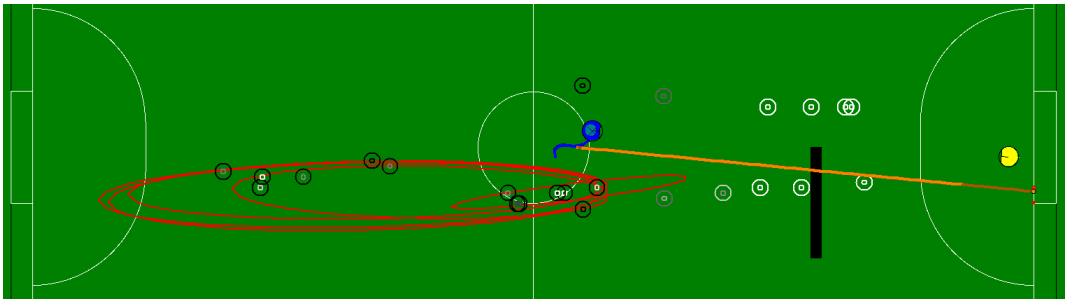



(c) Overshoot Goalie

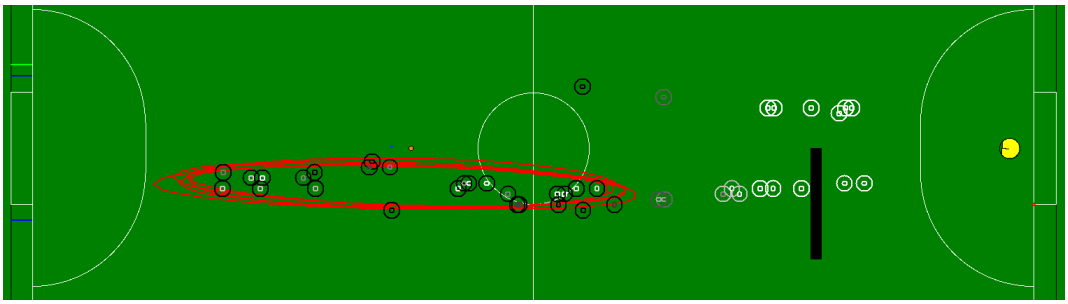
Figure 5.10: Moving average reward of the different online learning policies as a function of time, against different goalies. Shaded areas indicate 95% confidence intervals.



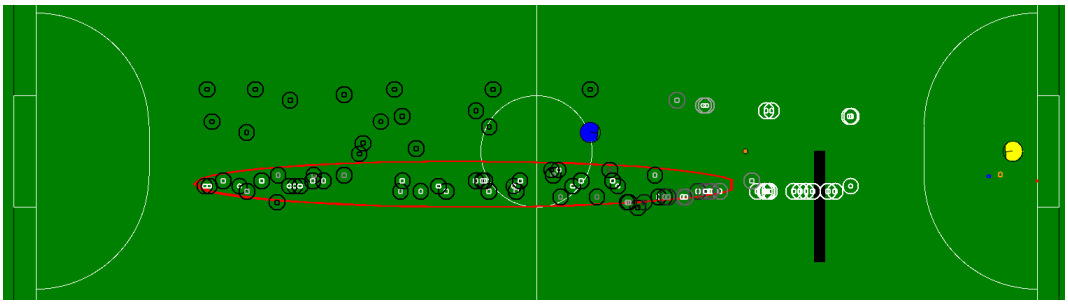
(a) 9 outcome observations



(b) 24 outcome observations



(c) 44 outcome observations



(d) 100 outcome observations

Figure 5.11: Example of the most anomalous region distribution (red ellipses) found during execution in the Obstructed Goalie domain. Concentric circles have the same meanings as those in Figure 5.9

**Overshoot Goalie.** Figure 5.12 shows an example of our RIM-UCB algorithm running on the Overshoot Goalie domain. Figure 5.10c shows the results of the policies on the Overshoot Goalie. The context-dependent inaccuracy in this domain shows a significantly smoother transition than the one in the other domains. Because of this smoothness, there is no significant difference among the performance of the three different methods; GP-UCB seems to obtain slightly larger reward, but it is within the margin of error.

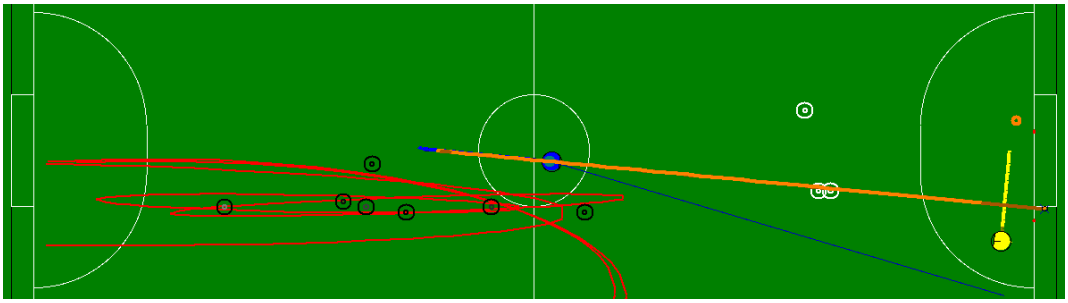
## 5.4 Chapter summary

This chapter presents an approach to enable robots to optimize their planning considering their own uncertainty about possible RIMs in their domains. To achieve this performance improvement, the approach first enables the robot to represent its uncertainty about RIMs; once the robot is able to represent this uncertainty, it can create plans that balance exploration –i.e., taking actions that reduce uncertainty about RIMs– with exploitation –i.e., taking actions that are known to yield high reward.

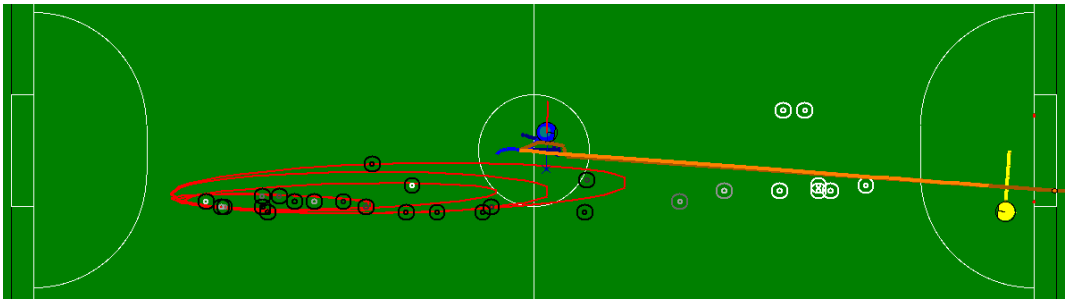
Section 5.1 provides the necessary background for this chapter. The section describes the Upper Confidence Bound algorithm for solving multi-armed bandit problems, and the Contextual Gaussian Processes Upper Confidence Bound in particular, which applies to domains with continuous state and action spaces, such as our domains of interest. In this algorithm, the robot represents its model, as well as its uncertainty about this model, using Gaussian Processes. Then, it uses the Upper Confidence Bound algorithm to effectively trade off exploration and exploitation.

Section 5.2 presents the proposed solution in technical detail. First, the approach enables robots to represent their uncertainty about possible RIMs in their domains. The robot approximates the probability distribution for each RIM with a collection of parametric regions of similar shapes, which are all consistent with the robot's outcome observations. Given this representation of the robot's uncertainty about each RIM, the section derives the expression for the robot's resulting estimated model, as well as its uncertainty about the model, defined over the robot's state-action space. Given the robot's estimate of the model and its uncertainty about it, the robot uses the Upper Confidence Bound algorithm to choose actions that effectively trade off model refinement and reward-gathering.

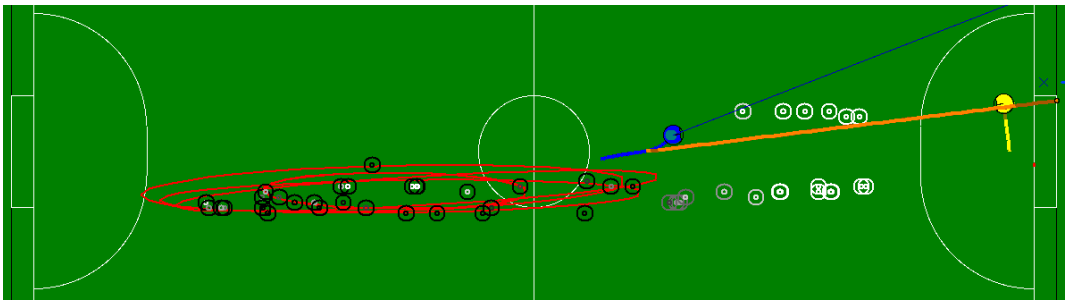
Section 5.3 demonstrates the value of the contributed approach empirically in a domain consisting of a soccer-playing robot that needs to repeatedly shoot on an inaccurately-modeled goalie to maximize its scoring. Experimental results show that, in domains with sharp discontinuities in the reward function, this thesis' RIM-based approach outperforms the state-of-the-art Gaussian-Processes-based approach to online learning. Furthermore, experiments show that combining the two approaches yields the best results.



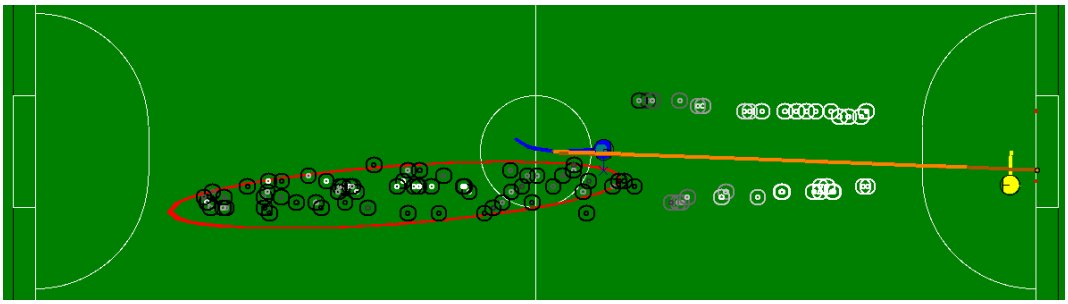
(a) 10 outcome observations



(b) 28 outcome observations



(c) 50 outcome observations



(d) 100 outcome observations

Figure 5.12: Example of the most anomalous region distribution (red ellipses) found during execution in the Overshoot Goalie domain. Concentric circles have the same meanings as those in Figure 5.9

# Chapter 6

# Related Work

We seek to enable robots to adapt at execution time, in the presence of RIMs in their state-action space, to optimize performance.

Section 6.1 contextualizes our work within the Execution Monitoring (EM) literature, which has historically addressed the problem of online detection of discrepancies between nominal planned execution and actual execution. However, unlike previous work in EM, we detect this off-nominal behavior as statistical *collective anomalies* in regions of the robot's state-action space.

Section 6.2 contextualizes our work within the Anomaly Detection (AD) literature, which has extensively studied the problem of finding spatial outliers. From them, we borrow the concept of contextual collective anomalies that enables our robots to detect subtle off-nominal behavior in particular RIMs; however, we adapt and apply this concept to online robot execution monitoring.

Finally, since we address the problem of online performance optimization in domains with model uncertainty, Section 6.3 relates our work to the Reinforcement Learning (RL) literature. We borrow existing approaches to this problem from the RL community, but adapt them to enable robots to explicitly reason about RIMs and their uncertainty about these RIMs.

In general, then, the biggest difference between our work and previous work is the joint application of concepts from these three fields, enabling robots to detect and reason explicitly about RIMs in their planning models.

## 6.1 Execution Monitoring

The problem of execution monitoring, also called Fault Detection and Identification (FDI), or Diagnosis (DX), depending on the community [18], is concerned with detecting, identifying and recovering from failures in execution. Execution monitoring is a well-established problem in various areas of scientific research, and the complex and unpredictable nature of robotics domains has led to increased exploration of execution monitoring in robotics [76]. Below, we situate our work in the context of the execution monitoring community. We only present related work in the field as it pertains to our work, but there are several surveys of execution monitoring as a general problem (e.g., [35, 40]) and specifically in robotics (e.g., [1, 76]). Throughout this section, we draw specific comparisons between our work and others in the field. However, the main distinguishing prop-

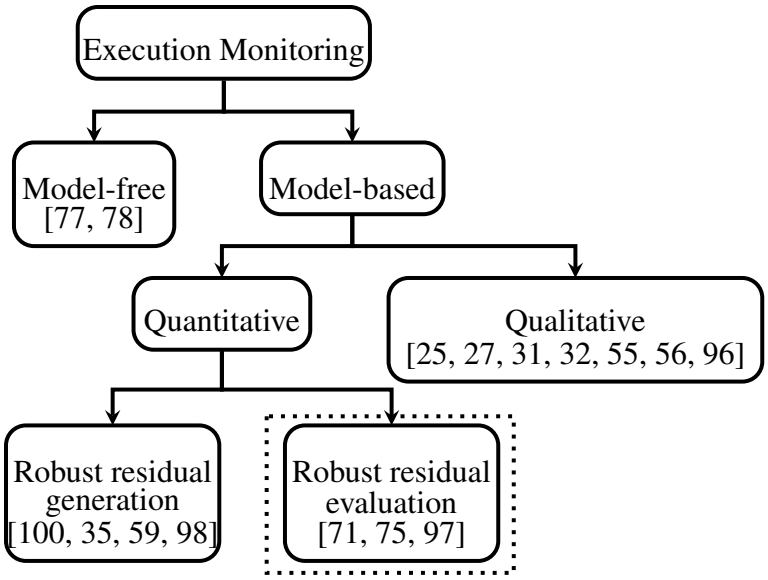



Figure 6.1: Rough taxonomy of execution monitoring methods inspired by previous survey work [39]. Our approach lies in the Robust residual evaluation leaf.

erties of our work are common to most of the approaches below: Execution monitoring research has focused on detection of faults characterized by *a single* unexpected observation, or *sequences* of unexpected observations; our work, on the other hand, focuses on detection of *regions* of the domain in which observations do not match expectations. Furthermore, our work seeks to enable robots to reason about their uncertainty about such expectations and act accordingly.

### 6.1.1 Taxonomy of execution monitoring approaches

Figure 6.1 shows a taxonomy of execution monitoring methods, strongly inspired by pervious survey work [39], and to the level of granularity appropriate for our work. At the highest level, execution monitoring can be divided into *model-based* methods, which achieve monitoring using models of the system, and *model-free* methods, which detect failures using only observed data. We are interested in adaptation in domains in which robots have access to partially accurate models of the world, and thus we focus on *model-based* monitoring; however, model-free methods have also been successfully applied in robotics [77, 78].

Model-based methods can be roughly separated into those that use *qualitative* models of the world, and those that use *quantitative* models of the world.

Qualitative model-based approaches use logic as the primary tool for monitoring. A common approach is to monitor the execution of plans using preconditions and effects of actions to check that the plan being executed is still consistent [27, 31, 96] or optimal [32]. In addition, temporal plan constrains may also be monitored [25, 55, 56]. Some of the challenges of qualitative model-based execution monitoring include reacting only to relevant changes in the world, and detecting plan failure as early as possible during execution.

Quantitative model-based approaches usually rely on the generation and analysis of numerical

*residuals*, rather than on logical elements. Residuals are differences between model-generated estimated values and the values observed during execution. In nominal execution, residuals are expected to be zero-mean vectors, and failures are detected as significant residual deviations from zero.

A comparative study of qualitative and quantitative approaches to model-based monitoring [18] has shown that there is very significant overlap between the expressive capabilities of both approaches. However, qualitative model-based methods are often used for higher level plan monitoring and diagnosing the reasons for these faults (e.g., [22, 21]), since they are based on the language of logic. On the other hand, quantitative model-based methods are generally used to monitor stochastic and continuous systems. Research efforts have attempted to reconcile these approaches into a joint framework that leverages the strengths of each [18, 34]. Furthermore, other approaches implicitly combine these by monitoring logical properties in a probabilistic manner [99]. In our work, we have approached the problem from a *quantitative* perspective, since we have focused on detection of failures in stochastic, continuous systems. The problem of incorporating our approach into qualitative methods is briefly discussed in Section 2.1.2.

Within the context of quantitative model-based monitoring, previous work [39] has argued that there are two ways to address stochasticity in the world: *Robust residual generation* methods, such as detection filters [98], robust observers [100], pairity relations [35] and Kalman filter methods [59], attempt to generate residuals that are minimally sensitive to noise and maximally sensitive to faults, while *robust residual evaluation* methods focus on using noisy residual measurements to detect faults in the system while minimizing false positive detections. Our work on failure detection has focused on residual evaluation, rather than generation, although incorporating robust residual generation methods would fit in straightforwardly with our approach. Section 6.1.2 compares our work more extensively to previous work that qualifies as robust residual evaluation methods.

### 6.1.2 Failure detection from noisy observations

We are interested in the problem of detecting RIMs given a list of stochastic contextual observations in which the robot's state $\boldsymbol{s}_t$ and action $\boldsymbol{a}_t$ provide the context, and the outcome $\boldsymbol{z}_t$ is the observation to be monitored. Most work in execution monitoring has focused on fault detection given a single observation $\boldsymbol{z}_t$ or a sequence of observations $[\boldsymbol{z}_i | i = 0, 1, \ldots, t]$, not using information about the context in which such residuals were observed. Several algorithms have been developed to address this problem, and their properties, such as speed of detection and detection power, have been extensively studied.

Two of the most studied algorithms, Sequential Probability Ratio Test (SPRT) [97] and Cumulative Sum control charts (CUSUM) [75], detect faults using thresholds on the likelihood ratio of residual observations, given a nominal model $\boldsymbol{\theta}^0$, and an alternative failure model $\boldsymbol{\theta}^1$. These algorithms can be very efficiently computed by maintaining an aggregate statistic $S_t$ and updating it in constant time with a new observation $\boldsymbol{z}_t$. However, they require prior knowledge about the fault model $\boldsymbol{\theta}^1$, or sets of models $\{\boldsymbol{\theta}^i\}$ for multiple fault detection [71]). In contrast, we are interested in problems in which the distribution of observations in the RIMs is unknown.

The Generalized Likelihood Ratio (GLRT) approach uses the maximum likelihood estimate of

| Method | Spatial | Contextual | Collective | Tractable | Incremental | Sparse |
|---|---|---|---|---|---|---|
| SPRT | | | ✓ | ✓ | ✓ | ✓ |
| CUSUM | | | ✓ | ✓ | ✓ | ✓ |
| GLRT | | | ✓ | ✓ | ✓ | ✓ |
| Spatial Outlier | ✓ | | | ✓ | | ✓ |
| Context Outlier | ✓ | ✓ | | ✓ | | ✓ |
| Image Anomaly | ✓ | ✓ | ✓ | | | |
| Spatial Scan | ✓ | ✓ | ✓ | | | ✓ |
| FARO/DMAPS | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Table 6.1: Comparison of different methods for anomaly detection and execution monitoring along various dimensions.

$\boldsymbol{\theta}^1$ for detection of faults with unknown parameters. Faults are again detected by thresholding a statistic $S_t$. However, $S_t$ requires computation that is no longer constant, but linear in $t$.

We have also worked on the problem of detecting failure from noisy sequential data [63]. Appendix C describes the work we conducted, in which we enabled the CoBot mobile robots to detect Motion Interference of various types using a Hidden Markov Model to capture the transitions between the different states of the robots.

In our approach, we seek to use similar statistical techniques as these well established methods. However, the key difference is that we need to detect inaccuracies that are constrained to particular regions of the robot's state-action space. Thus, we must use methods that consider the full contextual observation $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{z}_t)$, rather than just the outcome $\boldsymbol{z}_t$. Section 6.2 describes how we achieve this using techniques from Anomaly Detection.

## 6.2 Anomaly Detection

The Anomaly Detection community has extensively explored the problem of finding anomalies in spatial data. Table 6.1 summarizes the distinctions between our approach and previous work, as we describe more extensively below.

Previous survey work of Anomaly Detection research [14] generally classifies anomaly detection methods along four dimensions: nature of the data, availability of labeled data, type of output produced, and types of anomalies detected. For the sake of completeness, we situate our work along these four dimensions; we note, however, that the main distinguishing feature of our work is the types of anomalies detected.

**Data nature** We are primarily interested in problems in which both the state-action space and the expected observations could be *continuous variables*, but our algorithms extend naturally to problems in which the expected observations are *binary* variables. We have not addressed the problem of *categorical* or *binary state-action spaces*, but we believe extending our methods to such domains is a relevant problem, especially if trying to extend our methods to higher-level plan monitoring.

**Label availability** Throughout our work, we assume that a model of nominal execution is available to the robots, but that different types of unforeseen anomalies could occur. In the context of Anomaly Detection work [14], this is similar to a *semi-supervised* domain, in which labeled data only of nominal execution is available to the algorithms (we also assume that this data has been accurately captured in a model). This is in contrast to *unsupervised* domains in which no observations are available beforehand, and to *supervised* domains in which both nominal and anomalous behaviors have been previously observed.

**Output** Anomaly Detection algorithms may output *anomaly scores* or simply a *decision* regarding the existence of an anomaly. At different stages of our work, our algorithms may need to do either of these, depending on whether the task at hand is only to detect failure in execution, or to modify the behavior of robots based on the detection of failure. Our algorithms must also output the subspace of the domain in which an anomaly is detected.

**Type of Anomaly** This is the dimension along which our work differs from most other work, since most work on Anomaly Detection has focused on the problem of detecting single outlier observations during execution [14]. In contrast, we are interested in anomalies that are both *contextual* and *collective*. Below, we briefly describe each of these characteristics.

Given our model-based approach to execution monitoring, our augmented observations $\boldsymbol{z}'_i$ contain not only the observations $\boldsymbol{z}_i$ to be monitored, but also the *context* in which such observations were made –i.e., the state $\boldsymbol{s}_i$ and action $\boldsymbol{a}_i$ that led to such observation. This division of the data into two sets of dimensions with different semantics differs from most work on anomaly detection, in which data consists of a single vector to be compared to others. In our work, *spatial context* is given by the state-action point that produces each observation; however, non-spatial context, such as time sequences (e.g., [82, 93]), categorical profiles (e.g., [37, 9]), or graph connectivity (e.g. [89]), has also been explored in the literature. Spatial contextual anomalies have been studied as points $\boldsymbol{z}'_i = (\boldsymbol{x}_i, \boldsymbol{z}_i)$ for which the value of $\boldsymbol{z}_i$ is significantly different to that of the $k$ nearest neighbors of $\boldsymbol{x}_i$. Several algorithms have been developed to detect such anomalies [51, 57], and a formalization of the spatial outlier problem has been formulated [83]. Our work is different from these in that our anomalies are not only *contextual*, but also *collective*, in the sense that any single observation may not constitute an anomaly, and one must instead analyze groups of related observations to find anomalies.

Collective anomalies have been most widely studied in time series analysis, in which the goal is to detect sequences of observations that are anomalous [47, 46, 12]. Many of the approaches discussed in Section 6.1 could be classified as temporal collective anomaly detection algorithms in which the goal is to find sequences of observation that do not fit the expected model of execution. Our problem of interest is more general since our observations are related along multiple continuous dimensions (of which time may be one), rather than purely temporally related. Because of this, we need to develop different techniques that apply more generally, but which may not outperform techniques developed specifically for temporal anomaly detection.

The problem of detecting *spatial collective anomalies* has received significant attention from the Computer Vision (CV) community, as it can be used to detect anomalous regions of images, or simply to segment regions that stand out. Unfortunately, the algorithms developed for CV are not directly applicable to our problem: Images provide dense observations, in the sense that every pixel

in the image provides an observation that can be used for detection. Thus, CV techniques often exploit the lattice structure of image pixels to use graph-based algorithms to extract anomalous regions of the image (e.g., [30, 84]). This contradicts our goal of detecting anomalies from sparse observations over a continuous domain. Furthermore, since CV is highly focused on 2D or 3D images, CV algorithms are usually not applicable to problems of higher dimensions, such as our robotics problems.

Detection of spatial collective anomalies has also been addressed using spatial scan statistics [53]. Since our method is significantly influenced by work in this area, Section 6.2.1 situates our work in more detail within spatial scan statistics work.

### 6.2.1 Spatial scan statistics

The spatial scan statistic [53] is an approach for detecting regions of a multi-dimensional point process in which the number of observed points is significantly different from the number expected from a given model. As noted in previous work [53], such statistic has a wide range of applications, from forestry to astronomy; however, it has been most often studied in the context of early disease outbreak detection. The core idea of the algorithm is to search over a set of subspaces of the process domain to find the subspace $R^+$ that maximizes the following likelihood ratio $\Lambda(R)$:

$$\Lambda(R, \boldsymbol{Z}) = \frac{\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} P(\boldsymbol{Z}(R), \boldsymbol{\theta})}{P(\boldsymbol{Z}(R), \boldsymbol{\theta}^0)}, \tag{6.1}$$

where $\boldsymbol{\Theta}$ is the space of distribution parameters, $\boldsymbol{\theta}^0$ are the nominal distribution parameters, and $\boldsymbol{Z}(R)$ are the set of observations seen in $R$. This approach searches for the subspace $R^+$ most likely to be anomalous, after which it can perform inference to determine whether there exists an anomalous subspace of the domain, and where the anomaly is located. As originally developed [53], the algorithm for searching over the space of possible subspaces $R$ is not scalable to higher dimensions, because it performs an exhaustive search over the space of circular subspaces to find the one most likely to be anomalous; this search algorithm is feasible for the original context of the algorithm, in which only a 2-dimensional space had to be searched.

Our goal, with respect to anomaly detection, is to use the concepts of spatial scan statistics to create algorithms that allow runtime monitoring of robots. While more recent work has further developed the spatial scan statistics approach in several directions, none of the derived algorithms is directly applicable to runtime detection of anomalies of various shapes in continuous higher-dimensional domains. For example, exhaustive search of elliptical regions [54] or more efficient search for axis-aligned rectangles [70] have been proposed, but they do not scale well with the dimensionality of the domain. Graph-based approaches [28, 91] assume some connectivity between the data, which is not obviously obtained from sparse observations in higher dimensions. The Fast Subset Scan [68, 69], while efficient, limits its search to only subsets of regions of fixed radius around each observation.

Our approach to anomaly detection, described in Chapter 3, uses optimization to tractably search for anomalous regions, incrementally at runtime, in higher-dimensional robot domains.

| Method | MB | TO | SG | NM | MSM | OL |
|---|---|---|---|---|---|---|
| Min-error AL | ✓ | | ✓ | | | |
| Max-reward AL | ✓ | ✓ | ✓ | | | |
| Model+Correction | ✓ | | | ✓ | ✓ | |
| Model-free RL | | ✓ | ✓ | | | ✓ |
| $E^3$ / R-MAX | ✓ | ✓ | | | | ✓ |
| Function Approx MB RL | ✓ | ✓ | ✓ | | | ✓ |
| MMRL | ✓ | ✓ | ✓ | ✓ | | ✓ |
| RL-CD | ✓ | ✓ | ✓ | ✓ | | ✓ |
| RIM-UCB | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Table 6.2: Comparison of different active approaches to learning describing whether they are Model-Based (MB), are Task-Oriented (TO), provide Spatial Generalization (SG), discover New Models (NM), support Multiple Spatial Models (MSM), and learn OnLine during execution (OL). We contribute an online approach that support multiple spatial models.

## 6.3 Planning under model uncertainty

The RIM-UCB approach presented in Chapter 5 enables robots to make intelligent decisions despite inaccuracies in their models. Through RIM-UCB, robots can effectively trade off exploitation and exploration with respect to model inaccuracies. This exploration problem may be approached in two different ways, depending on the domain: In some domains, the robot can explore its domain with the sole purpose of refining its model to optimize future performance. In other domains, the robot does not have access to its task domain before its deployment, and thus must simultaneously optimize its performance while refining its models. Sections 6.3.1 and 6.3.2 describe related work pertinent to these two situations, respectively. Table 6.2 summarizes the relationships drawn below.

### 6.3.1 Offline active model learning

In many domains, robots can explore their domain with the sole purpose of optimizing future performance. For example, the CoBot robots might explore their buildings to get better models during the times when they have no scheduled tasks.

One strategy could consist of randomly exploring their domain, thus refining their models in random points. However, with real robots, it is often expensive to collect large amounts of data. Furthermore, we are particularly interested in scenarios in which only some subspaces of the domain have inaccurate modeling. For these reasons, we explore techniques that are more data efficient than random exploration.

In particular, Active Learning (AL) is concerned precisely with the problem of deciding where an agent should sample to improve its model of the world. AL has been employed in various robotics applications, such as learning the kinematics of a 2 DOF arm [16, 17], deciding which experiments a robot scientist should perform [48], and classifying visual input for a mobile robot [24].

However, the goal of these methods is to minimize future classification or regression error, while we are interested in active learning to optimize future task performance. Previous robotics work has explored AL to optimize the performance of a snake robot overcoming obstacles [92], and of a grasping hand [20]. The goal of these methods is to learn an approximation of the reward function from scratch; on the other hand, we seek a method to learn the boundaries of unmodeled modes of the system in which most of the domain is already well modeled.

The model plus correction approach [13] is a notable example of how robots might learn situation-dependent refinements to an existing model. In this approach, robots use their models to derive a policy, except for specific situations in which the model is known to be insufficient, in which case expert input replaces the model. The expert must decide when to give corrections, and decisions about whether to use the original model or the corrections are made by a predefined similarity threshold. In contrast, we seek algorithms in which the robots autonomously find situations in which their models are insufficient, actively try to correct their models, and autonomously generate the boundaries and characteristics of these discovered models.

### 6.3.2 Online active model learning

In some domains, it is not possible for the robot to explore its environment prior to task deployment. In these cases, the robot must then refine its model while trying to perform a task as close to optimally as possible. For example, the CMDragons soccer robots can only attempt to correct their models about opponents online during their game.

In such domains, Reinforcement Learning (RL) techniques address the issues of balancing exploration with exploitation for a robot that does not have a perfect model of its world. On the extreme end of non-complete modeling, Model-free RL approaches, such as Q-Learning [33] and policy gradient descent [90], have shown great performance in various robotics domains without explicitly modeling the world. Since they require no domain knowledge, model-free approaches are very general. While this generality is appealing and necessary in situations where modeling is impractical, model-free learning tends to be not very data-efficient and is not generalizable to different tasks within the same environment [2, 43].

In our domains of interest, we assume that a significant amount of domain knowledge is available; in fact, we address domains in which most of the domain is already well modeled, and our algorithms find specific subspaces in which the model is not sufficient. Therefore, model-based RL is more closely related to our work. Model-based RL approaches learn a model of the domain, and subsequently use such models for planning actions [43].

Various algorithms in model-based RL explicitly address the exploration vs exploitation problem. The $E^3$ [45] and R-MAX [10] algorithms guarantee near-optimal RL in polynomial time in the case of finite states and actions. However, they do not address the problem of generalization to unseen state-actions, and therefore are not applicable to our domains of interest, in which only sparse observations are available. Other algorithms for model-based RL have addressed the problem of generalization, usually via function approximation techniques [42, 72, 38]. In contrast to these algorithms, we seek to leverage knowledge about the fact that our model is accurate with the exception of a few unknown subspaces of the domain; this knowledge may enable our algorithms to achieve higher data efficiency than previous algorithms.

The problem of Multiple Model-based RL (MMRL) of the world has been explored by previous work, mostly for cases in which the number of different models is known a priori [15, 26]. Reinforcement Learning with Context Detection (RL-CD) [19] addresses a similar problem to ours, in that they seek to automatically generate new models throughout execution while doing RL. However, the switch among their models is a temporal one, in which only one model is active throughout space at a time. In contrast, we wish to infer spatial switches, and create new models accordingly.

The bottom line innovative contribution of our work is this discovery of new modes of behavior in state-action space, and the active characterization of the extent and behavior of these modes, with the end goal of task performance optimization.

# Chapter 7

# Conclusion

This chapter concludes the thesis with a summary of the work presented here, followed by a discussion of the implications of this work and of possible future work.

## 7.1 Thesis summary

**Thesis problem.** This thesis addresses the problem of enabling robots to act robustly, using sparse execution data, in the presence of subtle and context-dependent inaccuracies in their planning models. This problem arises in many complex robot domains in which it is infeasible for the robots to have globally accurate models at the beginning of their deployment. This infeasibility often arises because complex domains are far too large to explore exhaustively at training time, or because the robots cannot be trained in precisely the same domain as their deployment domain, or because computational restrictions require simple models that cannot capture the full complexity of the domain. These context-dependent model inaccuracies affect robot performance in particular regions of their state-action space, so this thesis focuses on detecting, correcting, and planning taking into account these regions.

**Thesis approach.** We present an approach that enables robots to explicitly reason about parametric Regions of Inaccurate Modeling (RIMs) in their state-action space. Approximating these inaccuracies as parametric regions enables robots to improve their execution robustness from sparse execution data by detecting these RIMs, correcting their models accordingly, and reasoning about the robot's uncertainty with respect to the existence, effect, and extent of these RIMs.

**Contribution 1: RIM-detection and correction in low-dimensional domains.** This work enables robots to detect and correct RIMs in their planning models online in low-dimensional domains. Our approach treats the problem of detection as an optimization problem, in which the robot searches for the parametric region of state-action space in which the contained data is most likely to have come from a different distribution than what its nominal model expected. Once the robot finds such region, it can use a statistical test to determine whether the unlikelihood of that data is statistically significant enough to be described as an anomaly. Given this detection, the

robot can proceed to apply a correction to its model that shifts the mean of the model's distribution by the most likely amount, according to the observed execution data. Empirical tests have shown that such corrections can significantly improve robot performance in complex tasks, such as robot soccer keepaway, with sparse execution data.

**Contribution 2: RIM-detection and correction in high-dimensional domains.** We extend the RIM-detection and correction approach to work effectively in high-dimensional domains. To achieve this, we assume that RIMs are intrinsically low-dimensional, but are embedded in a high-dimensional space; we argue that this is not a very restrictive assumption. This low-dimensional RIM assumption enables us to approach the problem as a search through low-dimensional subspaces of the robot's state-action space. This Feature Selection for RIM-detection (FS-RIM) approach uses best-first search to effectively search for the subspace of the domain that contains the most anomalous parametric region. We present three different heuristics for this best-first search, which trade off computational complexity and informativeness. Empirical evaluation on real CoBot data and on the golf-putting domain shows that FS-RIM vastly outperforms the performance of our low-dimensional approaches in high-dimensional domains.

**Contribution 3: Online learning in the presence of RIM-uncertainty.** We enable robots to make plans that take into account their uncertainty about the existence of context-dependent inaccuracies in their models. The robot has three types of uncertainty about these inaccuracies: uncertainty about whether these inaccuracies exist, uncertainty about their exact effect on the model, and uncertainty about the spatial extent of their effect in state-action space. We formulate mathematically how the robot can estimate each of these. In particular, to estimate its uncertainty about the spatial extent of model inaccuracies, we enable the robot to maintain an approximation to the distribution over possible RIMs. Representing these uncertainties enables the robot to estimate the expected value of its model's outcome predictions, as well as its own uncertainty about this estimate. Thus, the robot is able to use the well-established Upper Confidence Bound algorithm to effectively trade off exploration and exploitation in a Contextual Multi Armed Bandit problem. In the future, this formulation could be applied to other Reinforcement Learning problems.

**Related work.** Our approach brings together ideas from the fields of Execution Monitoring (EM), Anomaly Detection (AD) and Reinforcement Learning (RL) to enable robots to act robustly in the presence of model inaccuracies. From EM we take the idea of online monitoring to detect discrepancies between *expectations* generated by the robot's model of nominal execution, and stochastic *observations* received during actual execution. However, unlike most work in EM, we detect this off-nominal behavior as statistical *collective anomalies* in specific regions of the robot's state-action space. This concept of spatial collective anomalies, borrowed from AD literature, enables our robots to detect subtle deviations from nominal behavior from collections of observations that are contextually, and not necessarily temporally, correlated. Finally, we take ideas from the Reinforcement Learning community to enable robots to balance actions that refine their knowledge of RIMs with those that maximize their reward. We adapt the Upper Confidence Bound algorihtm to enable the robot to explicitly reason about RIMs.

**Evaluation in complex robot domains.** Aside from the theoretical and algorithmic contributions, this thesis contributes evaluation on various complex robot domains. We evaluate our approach on the CoBot service robot, which autonomously performs tasks in the Gates-Hillman Center at Carnegie Mellon University. This domain shows our approach working on an unconstrained environment in which the robot interacts on a daily basis with untrained humans. We also evaluate on the CMDragons soccer robot team. This domain evaluates our approach on a highly dynamic and adversarial domain. Importantly, it tests our approach in a domain that requires fast adaptation to an opponent that is inevitably inaccurately modeled in some situations. Finally, we evaluate our approach on the NASA spacecraft landing simulation. This domain demonstrates our approach detecting model inaccuracies of which the researchers were unaware, and on a model built entirely from a stream of data. Together, these domains demonstrate the general applicability of our approach to complex model-based robot domains.

## 7.2 Discussion and future work

This section discusses the implications of the work in this thesis, as well as possibilities for extending this work in the future.

### 7.2.1 Reasoning about discrete changes

This thesis enables robots to detect regions of their state-action space in which execution is anomalous, correct these anomalies, and plan accounting for them. While previous work has addressed the problem of estimating models and correcting them online, the bulk of that work consists of applying continuous corrections to the model.

While continuous corrections are often necessary, many of the inaccuracies in the real world are instead discrete. For example:

- A team of soccer robots must decide which opponents to mark, and they do so based on the state of the world. At some point, a very small change in the state of the world will cause a defender to switch from marking one opponent to marking another, who now poses a higher threat. This is an entirely discrete transition in the world –i.e., the robot does not smoothly change its marking behavior as a function of the world state. Similar discrete behavior transitions –e.g., play-switching, role-switching– occur at various levels of robot soccer and similar multi-robot domains.
- The CoBot robot has many unmodeled discrete transitions: the sun shining through the window, which affects its sensors, affects a very specific region of the building; the transition between soft carpet and rough tile is entierly discrete; the hallways of its building get crowded very quickly when class lets out. These discrete, or quasi-discrete transitions, may best be modeled as such.

Thus, we believe a valuable contribution of this thesis is to enable the robot to discover these discrete transitions in behavior as a function of its state-action space, and to model them separately from the rest of the space.

### 7.2.2 Explicit search for maximum anomaly

A key distinguishing attribute of this thesis, with respect to previous robotics work, is the explicit search for parametric regions of maximum anomaly value. This search enables robots to detect model inaccuracies from sparse execution observations. Even though throughout this thesis our algorithms search for ellipsoids of maximum anomaly, the approach can straightforwardly be applied to parametric regions of other shapes, or even to soft regions –i.e., regions with smooth boundaries whose smoothness can be controlled parametrically. It seems possible that even non-parametric approaches could borrow the idea of choosing hyper-parameters such that they maximize an anomaly value, rather than to optimize the fit to the observed data. We do not advocate for this choice in every domain, but we hypothesize that it might help in domains in which the goal is to find anomalies from sparse data.

### 7.2.3 Combining discrete and continuous approaches to model inaccuracies.

This thesis presents an online learning method that combines the parametric RIMs-based approach to model inaccuracies from this thesis with the non-parametric, Gaussian Processes (GP)-based approach to correcting models that is most frequent in the literature. Combining these two approaches enabled the robot's model correction to be robust in domains in need of smooth model corrections as well as those in need of discrete model corrections.

This work scratched the surface of combining such continuous, data-intensive approaches – e.g., Gaussian Processes (GPs)– with discrete, sparse-data-oriented approaches –e.g., RIMs. For example, in addition to combining the approaches simultaneously to achieve high continuous and discrete performance, one could combine them such that RIMs dominate corrections early in execution, when the data is sparser, while GPs dominate later in execution, when the data is dense. We believe such a combination could yield approaches that enable fast adaptation, as well as long-term fine-grained model estimates.

### 7.2.4 Time-dependent model inaccuracies

Inaccuracies in robot models often have a time-dependent component: the world may behave differently from expectations only for a specific time interval, or may do so in a periodic way. As long as time is included as one of the features in the robot's context space, our approach is able to detect and characterize model inaccuracies that happen within a specific time interval. Periodic inaccuracies, on the other hand, would only be correctly detected and characterized if the underlying information for the periodicity is included as one of the features in the robot's context space. For example, a particular corridor in the Cobot's building may be overcrowded on Wednesdays at 2 pm because a class ends at that time. Using our approach, the CoBot could only correctly characterize this periodic model inaccuracy if it had the day of the week and the time of day as features in its context space.

### 7.2.5 Separating inaccuracy detection from model correction

Throughout this thesis, we constrained the search for RIMs to those that have the effect of shifting the mean of the expected distribution in a particular region of context space. Upon detection, we presented an approach to model-correction that patches the nominal model with mean-shifted distributions in the detected RIMs. In some domains, the algorithm creators may have more specific knowledge about the form of the distribution, even in RIMs in the domain. As an example, it may be known that, even when the dynamics of a domain do not follow the expected distribution, they are still linear in nature. In such domains, it may be desirable to separate the detection step –which would continue as presented in this thesis– from the model correction step –which could consist, for example, of finding the best-fitting linear model for the detected RIM.

### 7.2.6 Theoretical guarantees of RIM-based online learning

We have demonstrated the value of our RIM-based online learning approach using empirical data from the robot soccer domain. However, one of the valuable attributes of the GP-based Upper Confidence Bound approach is its proven no-regret guarantees, given sufficient smoothness conditions about the robot's model. In the future, we would like to explore the problem of finding similar performance guarantees for our approach.

A challenge when addressing that problem is finding the correct set of assumptions about the underlying domain. Since this thesis explicitly targets domains with discrete transitions, we cannot adopt smoothness assumptions like the ones from the GP-based approach. However, some assumptions would be needed, since otherwise there could be a single point in state-action space that yields infintely more reward than every other point, and it is impossible for the robot to explore every pair of states and actions. One possible assumption could relate to the minimum size of RIMs in the domain: if RIMs are assumed to be larger than $\epsilon$, then the assumption-less problem above would not apply. Alternatively, one could make assumptions about the distribution of RIMs in the domain, and attempt to make probabilistic claims about the regret of the algorithm.

Whatever the assumptions may be, we believe providing theoretical guarantees would be a valuable contribution to enable further adoption of our approach.

### 7.2.7 Other types of reasoning about RIM-uncertainty

This thesis presents an approach to reason about the existence of RIMs in robot planning models, as well as an approach to represent the robot's uncertainty with respect to possible RIMs. With these tools, we have enabled our robots to effectively trade off exploration and exploitation in an online learning multi-armed bandit domain. One possible extension to our work is to use these tools in other types of domain that require reasoning about model uncertainty.

#### Bounded risk

Recent work [73, 74] has examined the problem of planning in stochastic systems using the notion of "bounded risk"–i.e., planning such that the probability of failure is bounded. Similarly, instead

of reasoning about stochastic but fully known models, the robot can reason about bounded risk in the context of stochastic and inaccurate models. In particular, the robot can take into account its uncertainty about RIMs during the risk allocation process.

#### Risk-sensitive planning

Similarly, reasoning about model uncertainty could affect the performance of risk-sensitive robots [49]. These are robots that care not only about the expected performance of their policies, but also the variance in performance. A risk-seeking robot prefers policies with higher variance in performance, while a risk-averse robot prefers policies with lower variance. In our domains, for example, a risk-seeking robot might intentionally try to visit locations of state-action space where it believes there might be a RIM of unmodeled high reward, even if there might not actually be a RIM there, which would result high negative reward.

#### Safe Exploration

The algorithms we have presented for exploration in search of advantageous RIMs does not explicitly account for safety. There may be certain regions of state-action space that the robot should never visit. This notion may be encoded into our algorithms in two different ways: first, the robot could have infinitely negative expected reward for such regions of state-action space; alternatively, the robot could explicitly encode some state-action pairs that are never allowed, and exclude those from its search for the optimal action to take.

Generally, reasoning about uncertainty is necessary in many robotics applications, and we have provided the tools needed to reason about uncertainty about context-dependent inaccuracies in robot planning models.

### 7.2.8 The value of parametric regions for human operators

One of the potentially significant advantages of detecting model inaccuracies as parametric regions, and one which we have left largely unexplored, is the value that human robot operators could derive from these parametric regions.

We, the algorithm designers, have experienced anecdotal evidence of this value multiple times throughout this thesis. A concrete example emerged from detecting RIMs in the moving ball interception model of the CMDragons [66]. In this domain, our model estimates the time that each robot needs to be able to intercept a moving ball. Using our RIM-detector on our own team, without any injected model inaccuracies, yielded a significant RIM: when our robot was aligned with the trajectory of the ball at the time the ball starts moving, it often took significantly more time than predicted to intercept the ball. When we found this anomaly, we delved into the robot logs to find its root cause. Because the RIM pointed to a specific set of situations –i.e., when our robot was aligned with the trajectory of the ball when the shot starts– finding the relevant data and analyzing them was relatively effortless. This analysis led the developers to discover an unintended behavior of the ball interception algorithm, which instructed the robot to stand still and wait for

the ball to arrive to it, instead of moving toward the moving ball to intercept it sooner. Correcting this unintended behavior led to significant passing performance in the CMDragons.

In general, because parametric regions are compactly represented, they often easier to understand for human developers than nonparametric methods. In the future, we would like to explore the hypothesis that reporting the parametric RIM to a human operator can lead to significantly faster discovery of the root source of anomalies in robot behavior, when compared to simply raising an execution alarm, or when compared to characterizing the anomaly in a nonparametric way.

### 7.2.9 Applying RIM-detection and correction to other complex domains

The evaluation domains in this thesis show the general applicability of reasoning about RIMs: the same ideas apply to the CoBot mobile service robot, the CMDragons soccer-playing robots, and the NASA spacecraft. More generally, these ideas apply to domains in which the robot has an observable but stochastic model of nominal behavior used for planning. Table 2.2 shows, for example, how our approach can be used for domains in which the robot models the behavior of the world as an MDP, as a factored MDP, and as a bandit reward function. We expect this formulation to be readily applicable to find and react to inaccuracies in various real-world domains, from autonomous cars' imperfect models of their motion dynamics and traffic paterns, to robot manipulators' imperfect models for approaching and grasping objects.

# Bibliography


[1] Gianluca Antonelli. A survey of fault detection/tolerance strategies for AUVs and ROVs. In *Fault diagnosis and fault tolerance for mechatronic systems: Recent advances*, pages 109–127. Springer, 2003. 89

[2] Christopher G Atkeson and Juan Carlos Santamaria. A comparison of direct and model-based reinforcement learning. In *International Conference on Robotics and Automation*, 1997. 96

[3] Peter Auer and Ronald Ortner. Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65, 2010. 70

[4] Richard Bellman. A markovian decision process. Technical report, DTIC Document, 1957. 2

[5] Joydeep Biswas. *Vector Map-Based, Non-Markov Localization for Long-Term Deployment of Autonomous Mobile Robots*. PhD thesis, The Robotics Institute, Carnegie Mellon University, December 2014. 7

[6] Joydeep Biswas, Juan P. Mendoza, Danny Zhu, Benjamin Choi, Steven Klee, and Manuela Veloso. Opponent-driven planning and execution for pass, attack, and defense in a multi-robot soccer team. In *Proceedings of International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, January 2014. 41, 124

[7] Joydeep Biswas and Manuela Veloso. Depth camera based indoor mobile robot localization and navigation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1697–1702. IEEE, May 2012. xi, 18, 19

[8] Joydeep Biswas and Manuela Veloso. Localization and navigation of the cobots over long-term deployments. *International Journal of Robotics Research*, 32(14):1679–1694, December 2013. 5, 39, 61

[9] Richard J. Bolton and David J. H. Unsupervised profiling methods for fraud detection. In *Proceedings of Credit Scoring and Credit Control VII*, pages 5–7, 2001. 93

[10] Ronen I. Brafman and Moshe Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, 3:213–231, March 2003. 96

[11] Brett Browning, James Bruce, Michael Bowling, and Manuela Veloso. STP: Skills, tactics, and plays for multi-robot control in adversarial environments. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 219(1):33–

52, 2005. 119

[12] Yingyi Bu, OTW Leung, AWC Fu, and EJ Keogh. WAT: Finding Top-K Discords in Time Series Database. *SDM*, 2007. 93

[13] Çetin Meriçli, Manuela Veloso, and H. Levent Akn. Improving biped walk stability with complementary corrective demonstration. *Autonomous Robots*, 32(4):419–432, 2012. 96

[14] Varun Chandola, A Banerjee, and V Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, pages 1–72, September 2009. 25, 92, 93

[15] Samuel P. M. Choi, Dit-Yan Yeung, and Nevin Lianwen Zhang. Hidden-mode markov decision processes for nonstationary sequential decision making. In *Sequence Learning - Paradigms, Algorithms, and Applications*, pages 264–287, London, UK, UK, 2001. Springer-Verlag. 97

[16] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Mach. Learn.*, 15(2):201–221, May 1994. 95

[17] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996. 95

[18] Marie-Odile Cordier, Philippe Dague, Franois Lvy, Jacky Montmain, Marcel Staroswiecki, and Louise Trav-Massuys. Conflicts versus analytical redundancy relations: a comparative analysis of the model based diagnosis approach from the artificial intelligence and automatic control perspectives. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 34(5):2163–2177, 2004. 89, 91

[19] Bruno C. da Silva, Eduardo W. Basso, Ana L. C. Bazzan, and Paulo M. Engel. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 217–224, New York, NY, USA, 2006. ACM. 97

[20] Christian Daniel, Malte Viering, Jan Metz, Oliver Kroemer, and Jan Peters. Active reward learning. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014. 96

[21] Johan De Kleer, Alan K Mackworth, and Raymond Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2):197–222, 1992. 91

[22] Johan De Kleer and Brian C Williams. Diagnosing multiple faults. *Artificial intelligence*, 32(1):97–130, 1987. 91

[23] Thomas Degris and Olivier Sigaud. *Factored Markov Decision Processes*, pages 99–126. John Wiley and Sons, Inc., 2013. 2

[24] Cristian Dima, Martial Hebert, and Anthony Stentz. Enabling learning from large datasets: Applying active learning to mobile robotics. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA 2004, April 26 - May 1, 2004, New Orleans, LA, USA*, pages 108–114, 2004. 95

[25] Patrick Doherty, Jonas Kvarnström, and Fredrik Heintz. A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems. *Autonomous Agents and Multi-Agent Systems*, 19(3):332–377, December 2009. 90

[26] Kenji Doya and Kazuyuki Samejima. Multiple model-based reinforcement learning. *Neural Computation*, 14:1347–1369, 2002. 97

[27] Richard J. Doyle, David Atkinson, and Rajkumar Doshi. Generating perception requests and expectations to verify the execution of plans. In *Proceedings of AAAI*, pages 81–88, 1986. 90

[28] Luiz Duczmal and Renato Assuncao. A simulated annealing strategy for the detection of arbitrarily shaped spatial clusters. *Computational Statistics & Data Analysis*, 45(2):269–286, 2004. 94

[29] Miroslav Dudík, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. Efficient optimal learning for contextual bandits. *CoRR*, abs/1106.2369, 2011. 117

[30] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, September 2004. 94

[31] Richard E Fikes, Peter E Hart, and Nils J Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3(0):251 – 288, 1972. 90

[32] Christian Fritz. Monitoring the execution of optimal plans. In *The 17th International Conference on Automated Planning and Scheduling (ICAPS) Doctoral Consortium*, Providence, Rhode Island, USA, September 2007. 90

[33] Chris Gaskett, David Wettergreen, and Alexander Zelinsky. Q-learning in continuous state and action spaces. In *Australian Joint Conference on Artificial Intelligence*, pages 417–428. Springer-Verlag, 1999. 96

[34] Sylviane Gentil, Jacky Montmain, and Christophe Combastel. Combining FDI and AI Approaches within Causal-Model-based Diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(5):2207 – 2221, 2004. 91

[35] J Gertler. Fault detection and isolation using parity relations. *Control Engineering Practice*, 5(5):653 – 661, 1997. 89, 90, 91

[36] David Haussler. Learning conjunctive concepts in structural domains. *Machine learning*, 4(1):7–40, 1989. 74

[37] Zengyou He, Xiaofei Xu, Joshua Zhexue Huang, and Shengchun Deng. Mining class outliers: concepts, algorithms and applications in {CRM}. *Expert Systems with Applications*, 27(4):681 – 697, 2004. 93

[38] Todd Hester and Peter Stone. TEXPLORE: Real-time sample-efficient reinforcement learning for robots. *Machine Learning*, 90(3), 2013. 96

[39] Inseok Hwang, Sungwan Kim, Youdan Kim, and Chze Eng Seah. A survey of fault detection, isolation, and reconfiguration methods. *Control Systems Technology, IEEE Transactions on*, 18(3):636–653, 2010. xiv, 20, 90, 91

[40] Rolf Isermann. Process fault detection based on modeling and estimation methods-a survey. *Automatica*, 20(4):387–404, July 1984. 89

[41] MV Ishkhanyan and AV Karapetyan. Dynamics of a homogeneous ball on a horizontal plane with sliding, spinning, and rolling friction taken into account. *Mechanics of Solids*, 45(2):155–165, 2010. 80

[42] Nicholas K. Jong and Peter Stone. Model-based function approximation for reinforcement learning. In *The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2007. 96

[43] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996. 96, 120

[44] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional bayesian optimisation and bandits via additive models. In *International Conference on Machine Learning*, pages 295–304, 2015. 78

[45] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Mach. Learn.*, 49(2-3):209–232, November 2002. 96

[46] Eamonn Keogh and Jessica Lin. Hot sax: Efficiently finding the most unusual time series subsequence. In *ICDM*, pages 226–233, 2005. 93

[47] Eamonn Keogh, Stefano Lonardi, and BY Chiu. Finding surprising patterns in a time series database in linear time and space. *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 550, 2002. 93

[48] Ross D. King, Kenneth E. Whelan, Ffion M. Jones, Philip G. K. Reiser, Christopher H. Bryant, Stephen H. Muggleton, Douglas B. Kell, and Stephen G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427:247–252, 15 January 2004. 95

[49] Sven Koenig and Reid G. Simmons. Risk-sensitive planning with probabilistic decision graphs. In *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR-94)*, pages 363–373, 1994. 104

[50] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324, 1997. 48

[51] Yufeng Kou and Chang tien Lu. Spatial weighted outlier detection. In *In Proceedings of SIAM Conference on Data Mining*, 2006. 93

[52] Andreas Krause and Cheng S Ong. Contextual gaussian process bandit optimization. In *Advances in Neural Information Processing Systems*, pages 2447–2455, 2011. 2, 12, 70, 72, 78

[53] M Kulldorff. A spatial scan statistic. *Communications in Statistics-Theory and methods*, 1997. 11, 21, 26, 28, 29, 35, 48, 49, 94

[54] Martin Kulldorff, Lan Huang, Linda Pickle, and Luiz Duczmal. An elliptic spatial scan statistic. *Statistics in medicine*, 25(22):3929–43, November 2006. 28, 94

[55] Solange Lemai and Flix Ingrand. Interleaving temporal planning and execution in robotics domains. In *Proceedings of AAAI*, pages 617–622, 2004. 90

[56] Steven James Levine. Monitoring the execution of temporal plans for robotic systems. Master's thesis, MIT, 2011. 90

[57] Chang-Tien Lu, Dechang Chen, and Yufeng Kou. Algorithms for spatial outlier detection. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, pages 597–, Washington, DC, USA, 2003. IEEE Computer Society. 93

[58] Patrick MacAlpine, Mike Depinet, and Peter Stone. UT Austin Villa 2014: RoboCup 3D simulation league champion via overlapping layered learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, volume 4, pages 2842–48, January 2015. 119

[59] Raman K Mehra and J Peschon. An innovations approach to fault detection and diagnosis in dynamic systems. *Automatica*, 7(5):637–640, 1971. 90, 91

[60] Juan Pablo Mendoza, Joydeep Biswas, Philip Cooksey, Richard Wang, Steven Klee, Danny Zhu, and Manuela Veloso. Selectively Reactive Coordination for a Team of Robot Soccer Champions. In *Proceedings of the Association for the Advancement of Artificial Intelligence Conference (AAAI)*, February 2016. 72

[61] Juan Pablo Mendoza, Joydeep Biswas, Philip Cooksey, Richard Wang, Steven Klee, Danny Zhu, and Manuela Veloso. Selectively reactive coordination for a team of robot soccer champions. In *Proceedings of the Association for the Advancement of Artificial Intelligence Conference (AAAI)*, to appear 2016. 122

[62] Juan Pablo Mendoza, Joydeep Biswas, Danny Zhu, Richard Wang, Philip Cooksey, Steven Klee, and Manuela Veloso. CMDragons 2015: Coordinated Offense and Defense of the SSL Champions. In *RoboCup 2015: Robot World Cup XVIX*. Springer, January 2016. 8, 124

[63] Juan Pablo Mendoza, Manuela Veloso, and Reid Simmons. Motion interference detection in mobile robots. In *Proceedings of IROS'12, the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Portugal, October 2012. 18, 92, 127

[64] Juan Pablo Mendoza, Manuela Veloso, and Reid Simmons. Focused optimization for online detection of anomalous regions. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, Hong Kong, June 2014. 18, 26, 33, 36, 48, 53, 55, 56, 115

[65] Juan Pablo Mendoza, Manuela Veloso, and Reid Simmons. Detecting and correcting model anomalies in subspaces of robot planning domains. In *Proceedings of International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Istambul, Turkey, May 2015. 11, 25, 26, 36, 119

[66] Juan Pablo Mendoza, Manuela Veloso, and Reid Simmons. Plan execution monitoring through detection of unmet expectations about action outcomes. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, Seattle, USA, May 2015. 18, 104

[67] Daniel B. Neill. *Detection of Spatial and Spatio-temporal Clusters*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2006. 27

[68] Daniel B. Neill. Fast subset scan for spatial pattern detection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(2):337–360, March 2012. 94

[69] Daniel B. Neill, Edward McFowland, and Huanian Zheng. Fast subset scan for multivariate event detection. *Statistics in medicine*, 32(13):2185–208, June 2013. 94

[70] Daniel B. Neill and Andrew W. Moore. Detecting significant multidimensional spatial clusters. *Advances in Neural Information Processing Systems*, 17:969–976, 2004. 94

[71] Igor V. Nikiforov. A generalized change detection problem. *IEEE Transactions on Information Theory*, pages 171–187, 1995. 90, 91

[72] Ali Nouri and Michael L. Littman. Multi-resolution exploration in continuous spaces. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1209–1216. Curran Associates, Inc., 2009. 96

[73] Masahiro Ono and Brian C Williams. An efficient motion planning algorithm for stochastic dynamic systems with constraints on probability of failure. In *AAAI*, pages 1376–1382, 2008. 103

[74] Masahiro Ono, Brian C. Williams, and Lars Blackmore. Probabilistic planning for continuous dynamic systems under bounded risk. *J. Artif. Int. Res.*, 46(1):511–577, January 2013. 103

[75] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1):100–115, 1954. 90, 91

[76] Ola Pettersson. Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems*, 53(2):73–88, November 2005. 89

[77] Ola Pettersson, Lars Karlsson, and Alessandro Saffiotti. Model-free execution monitoring in behavior-based mobile robotics. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*, pages 864 – 869, Coimbra, Portugal, 2003. 90

[78] Ola Pettersson, Lars Karlsson, and Alessandro Saffiotti. Model-free execution monitoring by learning from simulation. In *Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 505–511, Espoo, Finland, 2005. 90

[79] Joaquin Quiñonero Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *J. Mach. Learn. Res.*, 6:1939–1959, December 2005. 72, 117, 121

[80] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. 132, 135

[81] Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999. 33

[82] Stan Salvador, Philip Chan, and John Brodie. Learning states and rules for time series anomaly detection. In Valerie Barr and Zdravko Markov, editors, *FLAIRS Conference*, pages 306–311. AAAI Press, 2004. 93

[83] Shashi Shekhar, Chang-Tien Lu, Pusheng Zhang, S. Shekhar, C. T. Lu, and P. Zhang. A

unified approach to spatial outliers detection. *GeoInformatica*, 7:139–166, 2003. 93

[84] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, August 2000. 94

[85] Aleksandrs Slivkins. Contextual bandits with similarity information. *The Journal of Machine Learning Research*, 15(1):2533–2568, 2014. 70, 117

[86] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. Gaussian process bandits without regret: An experimental design approach. *CoRR*, abs/0912.3995, 2009. 117, 121

[87] Peter Stone, Gregory Kuhlmann, Matthew E. Taylor, and Yaxin Liu. Keepaway soccer: From machine learning testbed to benchmark. In Itsuki Noda, Adam Jacoff, Ansgar Bredenfeld, and Yasutake Takahashi, editors, *RoboCup-2005: Robot Soccer World Cup IX*, volume 4020, pages 93–105. Springer Verlag, Berlin, 2006. 40

[88] Peter Stone, Richard S. Sutton, and Gregory Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005. 8, 40, 119

[89] Jimeng Sun, Huiming Qu, Deepayan Chakrabarti, and Christos Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, pages 418–425, Washington, DC, USA, 2005. IEEE Computer Society. 93

[90] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pages 1057–1063, 1999. 96

[91] Toshiro Tango and Kunihiko Takahashi. A flexibly shaped spatial scan statistic for detecting clusters. *International journal of health geographics*, 4(1):11, 2005. 94

[92] Matthew Tesch, Jeff G. Schneider, and Howie Choset. Expensive function optimization with stochastic binary outcomes. In *ICML*, volume 28 of *JMLR Proceedings*, pages 1283–1291. JMLR.org, 2013. 96

[93] Ruey S Tsay, Daniel Peña, and Alan E Pankratz. Outliers in multivariate time series. *Biometrika*, 87(4), 2000. 93

[94] Manuela Veloso, Joydeep Biswas, Brian Coltin, and Stephanie Rosenthal. CoBots: Robust Symbiotic Autonomous Mobile Service Robots. In *Proceedings of IJCAI'15, the International Joint Conference on Artificial Intelligence*, Buenos Aires, Argentina, July 2015. 47

[95] Manuela Veloso, Joydeep Biswas, Brian Coltin, Stephanie Rosenthal, and Rodrigo Ventura. Cobots: Collaborative robots servicing multi-floor buildings. In *International Conference on Intelligent Robots and Systems (IROS)*, October 2012. 5, 36

[96] Manuela M. Veloso, Martha E. Pollack, and Michael T. Cox. Rationale-based monitoring for continuous planning in dynamic environments. In *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems*, pages 171–179, Pittsburgh, PA, June 1998. 90

[97] A. Wald. Sequential tests of statistical hypotheses. *Ann. Math. Statist.*, 16(2):117–186, 06

1945. 90, 91

[98] J. E. White and J. L. Speyer. Detection filter design: spectral theory and algorithm. *IEEE Transactions on Automatic Control*, AC-32(7):593–603, 1987. 90, 91

[99] Cristina M Wilcox and Brian C Williams. Runtime verification of stochastic, faulty systems. In *Runtime Verification*, pages 452–459. Springer, 2010. 91

[100] J. Wnnenberg and P.M. Frank. Sensor fault detection via robust observers. In Spyros Tzafestas, Madan Singh, and Gnther Schmidt, editors, *System Fault Diagnostics, Reliability and Related Knowledge-Based Approaches*, pages 147–160. Springer Netherlands, 1987. 90, 91

# Appendix A

# Computing $R^+$ in 1D

In a one-dimensional domain, the convex region (e.g., ellipse) of maximum anomaly $R^+$ can be computed in quadratic time, with respect to the number of data points, using dynamic programming. This algorithm exploits the fact that sufficient statistics for the anomaly measure of the union of two non-overlapping regions $R_1$ and $R_2$ can be computed in constant time once the sufficient statistics for each of them has been computed. Thus, starting with regions that surround each individual data point, the maximum anomaly region $R^+$ is found by merging adjacent regions and calculating their anomalies.

The anomaly measure of Equation 2.5 can often be computed from sufficient statistics of the data $\boldsymbol{Z}(\mathcal{R})$. For example, as shown in previous work [64], when trying to find a shift in the mean of normally-distributed observations, the logarithm of the anomaly $F(R) = \log \mathrm{anom}(R)$ is computed as:

$$F(R) = \frac{1}{2}\left(\sum_{\boldsymbol{x}_i \in R} \boldsymbol{\Sigma}_i^{-1} \Delta \boldsymbol{z}_i\right)^\top \left(\sum_{\boldsymbol{x}_i \in R} \boldsymbol{\Sigma}_i^{-1}\right)^{-1} \left(\sum_{\boldsymbol{x}_i \in R} \boldsymbol{\Sigma}_i^{-1} \Delta \boldsymbol{z}_i\right) \tag{A.1}$$

where $\boldsymbol{\Sigma}_i$ is the expected covariance of observation $\boldsymbol{z}_i$, and $\Delta \boldsymbol{z}_i$ is the deviation of observation $\boldsymbol{z}_i$ from its expected value, according to the nominal model. Thus, the statistics $S_{\Delta \boldsymbol{z}} = \sum_{\boldsymbol{x}_i \in R} \boldsymbol{\Sigma}_i^{-1} \Delta \boldsymbol{z}_i$ and $S_{\boldsymbol{\Sigma}} = \sum_{\boldsymbol{x}_i \in R} \boldsymbol{\Sigma}_i^{-1}$ are sufficient for computing $\mathrm{anom}(R)$. Furthermore, given the statistics for two non-overlapping regions $(S^1_{\Delta \boldsymbol{z}}, S^1_{\boldsymbol{\Sigma}})$ and $(S^2_{\Delta \boldsymbol{z}}, S^2_{\boldsymbol{\Sigma}})$, the statistics for the combination of their data is simply $S^{1+2}_{\Delta \boldsymbol{z}} = S^1_{\Delta \boldsymbol{z}} + S^2_{\Delta \boldsymbol{z}}$ and $S^{1+2}_{\boldsymbol{\Sigma}} = S^1_{\boldsymbol{\Sigma}} + S^2_{\boldsymbol{\Sigma}}$. Thus, it is possible to create a function $\mathrm{anom}(S)$ that computes the anomaly value from sufficient statistics of a region, and a function $\mathrm{merge}(S^1, S^2)$ that merges sufficient statistics from two non-overlapping regions; both of these run in constant time.

Given these sufficient statistics, Algorithm 8 describes the procedure of finding the region $R^+$ of maximum anomaly in 1D. The algorithm finds the range $R^+ = [\boldsymbol{x}^-_R, \boldsymbol{x}^+_R]$, where $\boldsymbol{x}^-_R, \boldsymbol{x}^+_R \in \mathbb{R}$. First, the observations are sorted along their context dimension (line 2). This ordering enables the dynamic programming to create a table (line 3) such that, by the end of the procedure, this table $T[i][j]$ contains sufficient statistics of the observations in the range $\left[\boldsymbol{Z}'_j, \boldsymbol{Z}'_{j+i}\right]$ to compute its anomaly value. This is achieved by first storing the statistics of each individual point in $T[0][j]$ (lines 5– 7), and then incrementally computing the statistics of larger regions by combining smaller ones (lines 8– 12). Finally, the most anomalous range can be computed by finding the statistics in

$T$ that produce the maximum anomaly value (lines 13–15). Figure A.1 illustrates the contents of the table for an example with 4 observations.

**Algorithm 8** Algorithm to find the region $R^+$ of maximum anomaly in a one-dimensional domain.
*Input:* Set of 1D contextual observations $\boldsymbol{Z}$, nominal model $\boldsymbol{\theta}^0$.
*Output:* The region $R^+$ that maximizes $\mathrm{anom}(R)$.

1: **function** FINDANOM1D( $\boldsymbol{Z} = [(\boldsymbol{x}_t, \boldsymbol{z}_t)|t = 0, \ldots, N]$, $\boldsymbol{\theta}^0$)
2: $\quad \boldsymbol{Z}' \leftarrow \mathrm{sort}(\boldsymbol{Z})$
3: $\quad$ ▷ 2D table stores stats of range $\left[\boldsymbol{Z}'_j, \boldsymbol{Z}'_{j+i}\right]$
4: $\quad T \leftarrow \mathrm{table}(|\boldsymbol{Z}'|, |\boldsymbol{Z}'|)$
5: $\quad$ **for** $j \in |\boldsymbol{Z}'|$ **do**
6: $\quad\quad T[0][j] \leftarrow \mathrm{anomStats}(\boldsymbol{z}_j)$
7: $\quad$ **end for**
8: $\quad$ **for** $i \leftarrow 1$ to $|\boldsymbol{Z}'|$ **do**
9: $\quad\quad$ **for** $j \leftarrow 0$ to $|\boldsymbol{Z}'| - i$ **do**
10: $\quad\quad\quad T[i][j] \leftarrow \mathrm{merge}(T[i-1][j], T[0][i+j])$
11: $\quad\quad$ **end for**
12: $\quad$ **end for**
13: $\quad (i^*, j^*) \leftarrow \arg\max_{(i,j)} \left[\mathrm{anom}\left(T[i][j]\right)\right]$
14: $\quad R^+ \leftarrow \left[\boldsymbol{Z}'_{j^*} - \epsilon, \boldsymbol{Z}_{j^*+i^*} + \epsilon\right]$
15: $\quad$ **return** $R^+$
16: **end function**

| | | | |
|---|---|---|---|
| $S([\boldsymbol{x}_0, \boldsymbol{x}_3])$ | | | |
| $S([\boldsymbol{x}_0, \boldsymbol{x}_2])$ | $S([\boldsymbol{x}_1, \boldsymbol{x}_3])$ | | |
| $S([\boldsymbol{x}_0, \boldsymbol{x}_1])$ | $S([\boldsymbol{x}_1, \boldsymbol{x}_2])$ | $S([\boldsymbol{x}_2, \boldsymbol{x}_3])$ | |
| $S([\boldsymbol{x}_0, \boldsymbol{x}_0])$ | $S([\boldsymbol{x}_1, \boldsymbol{x}_1])$ | $S([\boldsymbol{x}_2, \boldsymbol{x}_2])$ | $S([\boldsymbol{x}_3, \boldsymbol{x}_3])$ |

Figure A.1: Dynamic programming procedure to obtain the range of maximum anomaly value in 1D. Each entry contains the sufficient statistics $S[\boldsymbol{x}_i, \boldsymbol{x}_{i+j}]$ required to compute $\mathrm{anom}([\boldsymbol{x}_i, \boldsymbol{x}_{i+j}])$.

# Appendix B

# Online Learning of Robot Soccer Free Kicks using Gaussian Processes

Online learning is an appealing and challenging problem in the domain of autonomous robot soccer. Online adaptation is essential to optimize performance against previously unknown opponents with varied strategies. However, the planning space of the team is extremely large, and the robots only have a few minutes of execution to adapt. This chapter focuses on online learning for *offensive free kicks* –i.e., free kicks taken by our team from the opponent's half of the field– for which we would want to find weaknesses in the opponent's marking, leading to repeated scoring.

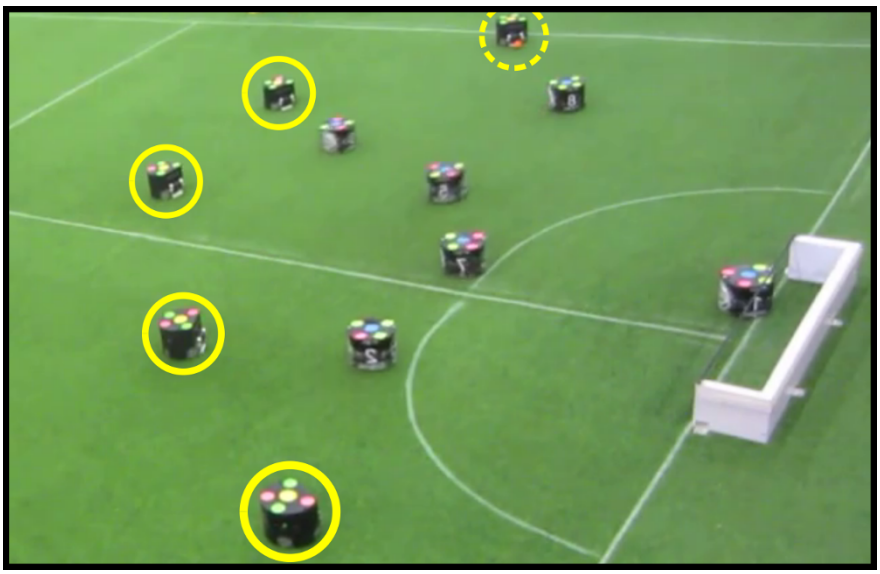
In a game of the RoboCup Small Size League (SSL) [1] soccer, there are around 10 to 20 offensive free kicks per game, making it necessary to adapt from sparse data. To this end, we approach the problem as a multi-armed bandit problem [85], in which the team must choose among a small finite set of pre-computed *Free Kick Plans* (FKPs) as their actions, which yield a reward of 1 if they score a goal within a short time after the free kick –e.g., the FKP of Figure B.1– or 0 otherwise. The effectiveness of different FKPs heavily depends on the location from which the free kick is taken, so we approach the problem as a contextual multi-armed bandit problem [29] with a metric context [85].

Our proposed approach to learning is for the team to model an estimate of the reward function using Gaussian Process-based regression [79] for each FKP, and then choosing the next action to be the one that maximizes the Upper Confidence Bound (UCB) acquisition function [86], thus guaranteeing a no-regret learning process. This approach is closely related to the Contextual Gaussian Process Upper Confidence Bound (CGP-UCB) approach described in Section 5.1.

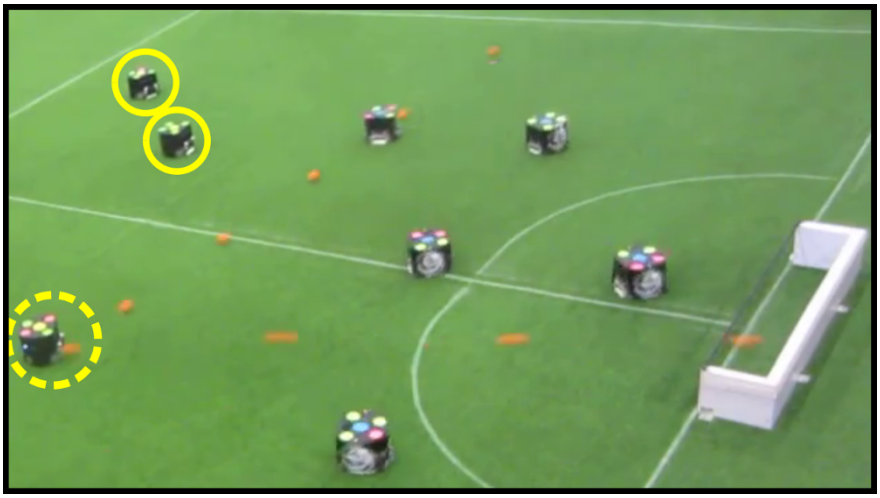
We evaluate our online learning algorithm using a physics-based SSL soccer simulation. We demonstrate the effectiveness of the algorithm against three realistic defending teams, each with different weaknesses and strengths.

Concretely, this chapter presents three contributions: (a) A framework for modeling the problem of online learning of free kicks as a contextual multi-armed bandit problem, (b) An algorithm for addressing this problem, and (c) empirical evidence for the effectiveness of the algorithm.

[1] wiki.robocup.org/wiki/Small_Size_League

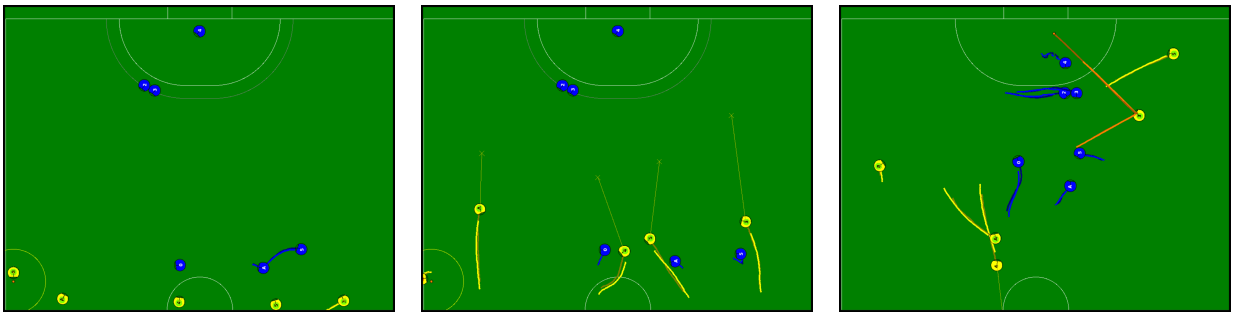


(a) FKP setup (dashed circle robot passes ball)



(b) FKP execution and scoring (dashed circle robot shoots ball)

Figure B.1: Free Kick Plan (FKP) successfully executed at RoboCup 2015. Yellow circles show our team's robots. We present an algorithm for learning effective FKPs online.



(a) Setup: Kicker on the left, others at $p_i^s$

(b) Move to target locations (thin yellow lines)

(c) Pass and score (ball path in orange)

Figure B.2: Plan $\mathrm{fk}_5$ at RoboCup: Kicker (left yellow) passes as teammates (yellow) charge to the opponent's (blue) goal (top).

## B.1 Background: Robot soccer and free kicks

The problem of reinforcement learning has been addressed in several robot soccer applications, such as simulation of keepaway [88] and layered learning of soccer behaviors [58]. However, online learning has not been applied nearly as frequently. Online learning has been applied to adjust weights of different formations depending on their success [11], and to improve the models of passing success against an unknown opponent [65]; neither of these explicitly addressed the exploration vs. exploitation problem. One of the reasons for the lack of research on online learning for robot soccer is the difficulty of the problem: since soccer games only last a few minutes, and each opponent is only encountered once, robots must learn from very sparse data in a very high-dimensional domain. This is the reason why our work focuses on learning from a small set of actions in the more specific domain of robot soccer free kicks.

In the SSL, free kicks are a method of restarting the game after an infraction, or after the ball has left the field. A free kick is awarded to one of the teams at the closest legal location to the infraction. Until the kicking team restarts play by touching the ball, all robots from the opposing team must maintain a distance of at least $50cm$ from the ball. In this chapter, we focus on online learning for offensive free kicks, since they provide semi-controllable initial conditions for our plans –the ball is stationary, and we can choose where to position our robots – and they provide scenarios with a relatively high chance of scoring, especially since we focus on offensive free kicks on the opponent's half of the field.

## B.2 Free kick planning as a bandit problem

The full planning space of offensive free kick plays consists of continuous and high-dimensional state and action spaces. The full state space consists of more than 80 physical dimensions, including the position, orientation, and velocities of the 12 robots and the ball, plus the state of the game and the internal state of each team. The action space is also high-dimensional, as robots

can move arbitrarily within physical limitations, and they can execute long sequences of passes, dribbling and shooting. We propose to make online learning feasible from sparse observations by greatly reducing the planning space and modeling the problem as a contextual multi-armed bandit problem.

**State Space.** Offensive free kicks allow our team to control the initial conditions of the world to a large extent: the ball is stationary at position $p^b$, and we can place our robots at arbitrary feasible initial conditions. Furthermore, we assume that the adversary does not change its behavior throughout the game, and thus the opponent's reactions to our plans are relatively repeatable as well. We therefore reduce the planning state $\boldsymbol{s}$ to the two-dimensional initial ball location $\boldsymbol{s} = p^b$ from which the free kick is taken. The effectiveness of different plans highly depends on $p^b$.

**Action Space.** To reduce the size of the action space, we define a set of *Two-Step Free Kick Plans* (2FKPs) that consist of the following sequence: First, every robot $\rho_i$, excluding the goalie $\rho_g$ and the free kick taker $\rho_k$, proceeds to a setup location $p_i^s$; then, the robots proceed to final target locations $p_i^f$, while $\rho_k$ passes to the best potential target robot $\rho^*$, at the best computed location $\hat{p}_i^f$ within a fixed radius of $p_i^f$. Given a team of $N_\rho$ robots, then, a 2FKP can be expressed as a vector $\boldsymbol{a}$ of length $2(N_\rho - 2)$ of locations $p_i^s$ and $p_i^f$ for each potential receiver. 2FKPs are expressive enough to contain dynamic plans with $p_i^s \neq p_i^f$, yet simple enough to enable a bandit formulation, rather than more general reinforcement learning [43] over long sequences of actions.

We further restrict the set of possible free kicks –because of our goal of learning from sparse observations– to a finite set of 2FKPs containing $N_{\boldsymbol{a}}$ elements: $\boldsymbol{A} = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_{N_{\boldsymbol{a}}}\}$. Thus, during each offensive free kick, the team must choose among $N_{\boldsymbol{a}}$ possible actions.

Figure B.2 illustrates an example of a 2FKPs being executed during RoboCup 2015. In this $\mathrm{fk}_5$ plan, all the robots except for $\rho_g$ and $\rho_k$ spread around the midfield for the setup (Figure B.2a), and then proceed to charge forward to locations around the opponent's goal (Figure B.2b) to receive a pass and shoot (Figure B.2c).

**Reward Function.** We seek to maximize the number of goals scored during offensive free kicks, and thus we specify the rewrad function $r$ as $r = 1$ if our team scores within time $t_{FK}$ of the kick, or $r = 0$ otherwise. Time $t_{FK}$ is a threshold indicating an approximate time after which scoring is no longer attributed to the chosen 2FKP; in our work, $t_{FK} = 10s$.

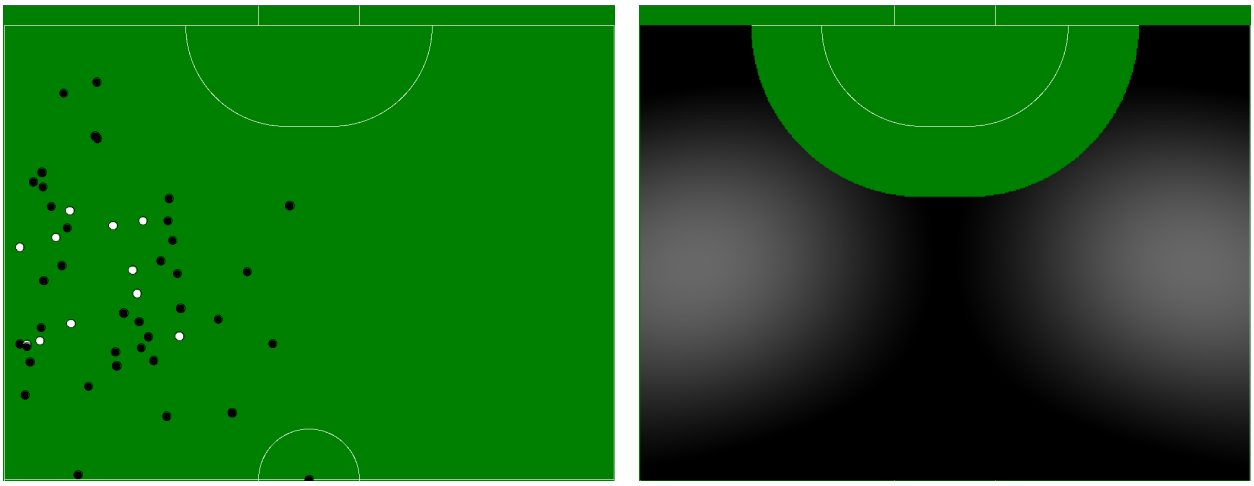
## B.3 Online learning over a set of 2FKPs

We enable online learning by (1) modeling an approximation to the expected reward function $\bar{r}(\boldsymbol{s}, \boldsymbol{a})$ as a function of the state of the world $\boldsymbol{s}$ and the chosen action $\boldsymbol{a}$, and (2) appropriately choosing actions that intelligently trade-off exploitation of known good actions, with exploration of actions with uncertain results.

**Approximating $\bar{r}$ using Gaussian Processes** We approximate $\bar{r}$ using Gaussian Processes (GPs)-based regression [79]. We do this through a bank of GPs $\{\mathrm{GP}^{\boldsymbol{a}_1}, \mathrm{GP}^{\boldsymbol{a}_2}, \ldots, \mathrm{GP}^{\boldsymbol{a}_{N_{\boldsymbol{a}}}}\}$, one for each available action. We start by approximating $\bar{r}$ by a prior function $\hat{r}_0^{\boldsymbol{a}}(\boldsymbol{s})$; then, we use subsequent observations $(\boldsymbol{s}^i, \boldsymbol{a}^i, r^i)$ of free kick states $\boldsymbol{s}^i$, chosen action $\boldsymbol{a}^i$, and resulting reward $r^i$ to update this estimate online. Thus, the approximating function $\hat{r}(\boldsymbol{s}, \boldsymbol{a})$ is given by:

$$\hat{r}(\boldsymbol{s}, \boldsymbol{a}) = \mathrm{GP}^{\boldsymbol{a}}\left(\boldsymbol{s} | \hat{r}_0^{\boldsymbol{a}}, (\boldsymbol{s}^1, r^1), \ldots (\boldsymbol{s}^n, r^n)\right),$$

where each pair $(\boldsymbol{s}^i, r^i)$ is an observation in which $\boldsymbol{a}^i = \boldsymbol{a}$. Our algorithms can query this bank of GPs to find the expected value $\mu(\boldsymbol{s}, \boldsymbol{a})$ and the variance $\boldsymbol{\Sigma}(\boldsymbol{s}, \boldsymbol{a})$ of the estimate expected reward function $\hat{r}(\boldsymbol{s}, \boldsymbol{a})$. Figure B.3 shows the expected value function $mu$ estimated from success and failure data of a particular 2FKP.



(a) Reward: 0 (black), 1 (white)

(b) Reward function estimate

Figure B.3: Example expected reward estimate over valid free kick locations. We only show samples from one half of the field length-wise, as our implementation assumes symmetry.

**Upper Confidence Bound Online Learning** Given estimates for the mean and variance of the modeled reward function, we use the Upper Confidence Bound (UCB) algorithm to choose the next action $\boldsymbol{a}$, given state $\boldsymbol{s}$:

$$\boldsymbol{a}(\boldsymbol{s}) = \underset{\boldsymbol{a}' \in \boldsymbol{A}}{\arg\max} \left[\mu(\boldsymbol{s}, \boldsymbol{a}') + \beta\sigma(\boldsymbol{s}, \boldsymbol{a}')\right], \tag{B.1}$$

where $\beta$ is a parameter that controls the level of exploration vs. exploitation in the algorithm. The UCB algorithm has been shown to be a no-regret algorithm [86], guaranteeing that difference between the reward of our chosen action and the reward of the optimal action grows sublinearly as the team learns which action to take.

Algorithm 9 illustrates the process of online learning of FKPs during a game. If the team must select a free kick plan to execute, it uses Equation B.1 to choose the next action. At the end of the play, the team sees a reward of either 0 or 1, depending on whether it scored a goal, and adds the observation to the right GP.

**Algorithm 9** Free kick plan online learning procedure.

**procedure** LEARNFREEKICK(game_state)
  **if** game_state = select_free_kick **then**
    $\boldsymbol{s}_i \leftarrow p^b$
    $\boldsymbol{a}_i = \arg\max_{\boldsymbol{a} \in \boldsymbol{A}} [\mu(\boldsymbol{s}_i, \boldsymbol{a}) + \beta\sigma(\boldsymbol{s}_i, \boldsymbol{a})]$
  **end if**
  **if** game_state = play_end **then**
    $r_i \leftarrow 1$ if goal, $0$ otherwise
    Add $(\boldsymbol{s}_i, r_i)$ to $\mathrm{GP}^{a_i}$
  **end if**
**end procedure**

## B.4 Experimental results

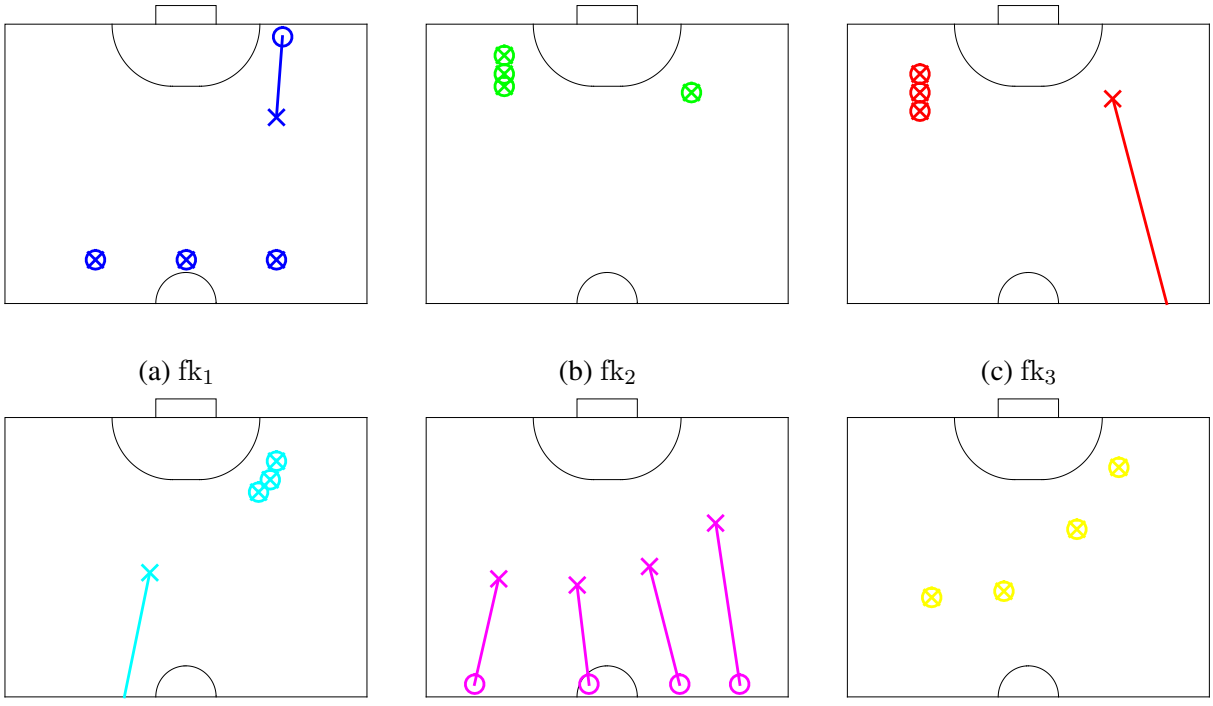
The goal of this work is to achieve advantageous FKP adaptation online during RoboCup games. In fact, during RoboCup 2015, our team did perform such adaptation during real games, and it won the SSL tournament. However, it is very difficult to accurately evaluate the amount of credit that the online learning of free kicks deserves in that victory due to (i) lack of ground truth, (ii) small number of games, and (iii) the large proportion of the games that does not involve free kicks, such as offense coordination during regular gameplay [61]. Thus, we instead present a controlled experimental evaluation of our algorithm.

We evaluate the proposed FKP modeling and action selection algorithm on a PhysX-based simulation of a robot soccer game of the SSL. We equipped the team with 6 different 2FKPs, illustrated in Figure B.4. Here, we describe how we obtained the true expected reward function of each 2FKP, and the results of evaluating our algorithm against three different defending teams.

### B.4.1 Defense teams

We evaluated our online learning algorithm against three different defense teams. For each defense, the closest robot to the goal is the goalie, who intercepts incoming shots, and the closest robot to the ball attempts to gain control of it by driving to intercept it optimally. One or two of the remaining robots, depending on how many are needed, block open angles from $p^b$ to the goal.

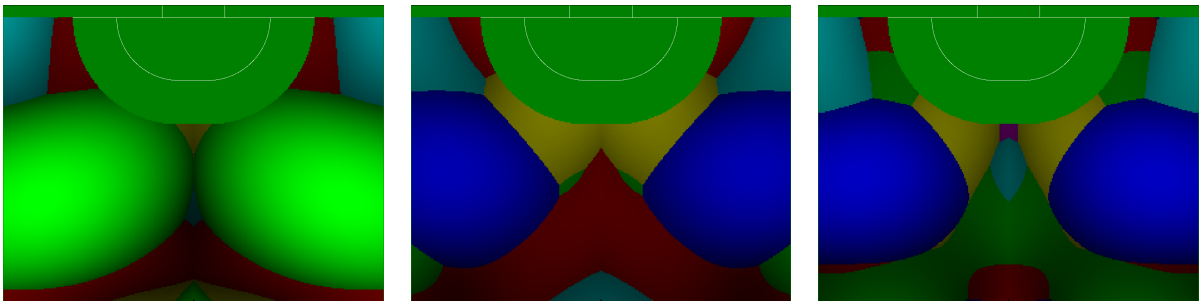
The remaining robots are assigned to block threats on the goal from the most threatening opponents. This evaluation of threats is different for each defense. The *Time Defense* ranks opponent threats based on how long it would take each robot located at location $p^i$ to receive a pass from $p^b$ to $p^i$, assuming a ball speed of $5m/s$, and then shoot on the goal, assuming the maximum legal shot speed of $8m/s$, prioritizing robots with shorter times. The *Angle Defense* ranks opponent threats based on the size of the open angle they have on the goal from their location $p^i$, prioritizing robots with wider shooting angles. The *Combined Defense* ranks opponent threats based on a combination of the measures of Time and Combined, using the open angle measure of Angle only if the robot's open angle is smaller than a threshold $\phi_{\max}$. If multiple robots have an open angle wider than $\phi_{\max}$, they are compared based on the time measure of Time. This realistic defense threat evaluation is



(a) $\text{fk}_1$ (b) $\text{fk}_2$ (c) $\text{fk}_3$

(d) $\text{fk}_4$ (e) $\text{fk}_5$ (f) $\text{fk}_6$

Figure B.4: Available free kick plans. Circles mark initial receiver locations $p_i^s$, while Xs mark their target locations $p_i^f$. Plots assume that $p^b$ is in the left half of the field; otherwise, the plans are mirrored about the $x$-axis.



(a) Time defense (b) Angle defense (c) Combined defense

Figure B.5: Optimal action map. Different colors represent different FKPs, consistent with the colors of Figure B.4. Color intensity, between 0 and 1, is the nonlinear function $\sqrt{r}$ for ease of visualization.

the actual evaluation used by the CMDragons SSL team [6], who are the current champions of the SSL [62].

### B.4.2 Online learning evaluation

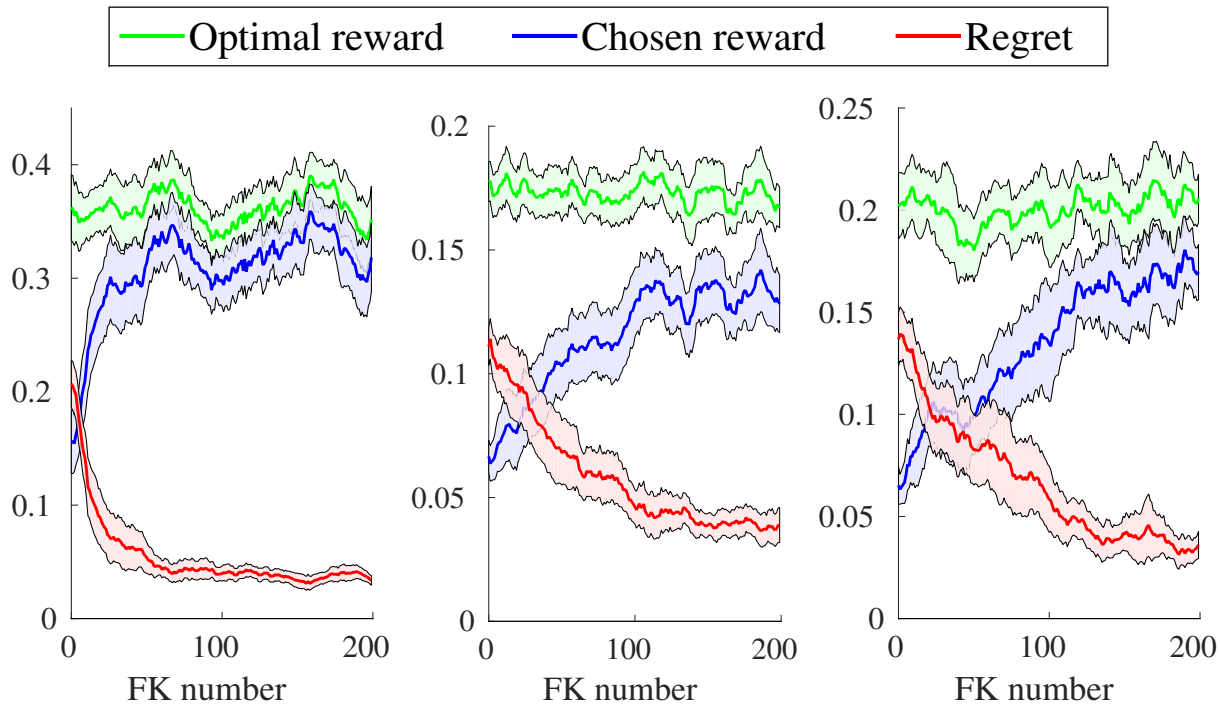
To evaluate our algorithm against the 3 different teams above, we first obtain an accurate estimate of the expected reward function $\bar{r}^{\boldsymbol{a}}$, for each action $\boldsymbol{a}$, and for each of the 3 teams. Then, using online learning, we evaluate the evolution of the expected regret in time.

**True Expected Reward.** For each 2FKP, we ran extensive simulation free kicks from a fine grid of locations $p^b$, and used GPs to model the expected reward. This function approximation becomes increasingly accurate as we run more free kicks; we obtained the true reward function with $\sim 1000$ free kicks for each 2FKP. Figure B.5 illustrate the resulting expected reward function of the optimal action $\boldsymbol{a}^*$ for each free kick location $p^b$ on the field.

**Online Learning Performance.** We evaluated our algorithm against each defense by conducting sequences of free kicks from the forward-left quadrant as during training, but using a random sequence instead of a grid sequence. Furthermore, the team selected their action according to our online learning algorithm. For each chosen action, we measured the expected regret $R$ as:

$$R(\boldsymbol{a}^i|\boldsymbol{s}^i) = \max_{\boldsymbol{a}' \in \boldsymbol{A}} \left[ \bar{r}^{\boldsymbol{a}'}(\boldsymbol{s}^i) - \bar{r}^{\boldsymbol{a}^i}(\boldsymbol{s}^i) \right] \tag{B.2}$$

Figure B.6 illustrates the average evolution of regret for each defense team, along with the expected reward of the optimal action and the expected reward of the chosen action. The optimal reward, as we average over more learning episodes, converges toward a horizontal line, different for each of the three defenses. The regret decreases significantly even within the first 10-20 free kicks, and for the Time defense it nearly reaches 0 within that time; this indicates that our algorithm could significantly improve the performance of our team during RoboCup games.




(a) Time defense (b) Angle defense (c) Combined defense

Figure B.6: Results of online learning against three defense teams. Average regret decreases quickly for each of the defenses. Shaded areas indicate a 95% confidence interval for the value of each function.

# Appendix C

# Detection of Anomalous Motion Data Sequences in the CoBots

This appendix describes our previous work [63] on detection of anomalies in sequences of observations. The sequential nature of the anomalies contrasts with the approach throughout this thesis, in which time is only one of the dimensions taken into account when finding correlations among the data.

## C.1 The Motion Interference detection problem

This work focuses on detecting anomalies in the motion of the CoBot robots. We focus on the following types of motion interference, as they are the most relevant to the CoBot's execution, though we believe our algorithm is more generally applicable to motion interference.

**Collision with partially detectable objects (*collision*)** While the fairly reliable perception mechanisms of the CoBots, combined with obstacle avoidance algorithms, can provide successful local navigation the vast majority of the time, there are some obstacles that are either undetectable or only partially detectable to the sensors. Transparent obstacles are particularly challenging for such light-based sensors, but opaque obstacles can also be only partially detectable. For example, the top of the table shown in Figure C.1a is too tall for either the Kinect or the laser range-finder to perceive it. While the robot avoids the leg of the table successfully, it cannot detect and avoid the much larger full width of the table, leading to potential collisions.

**Being held by a person (*hold*)** In some situations, a person may want to stop the CoBot's motion, and though the CoBots have emergency stop buttons, a person's first reaction when needing to stop the robot may reasonably be to directly stop it by holding it, as shown in Figure C.1b. Furthermore, the effect of being suddenly grabbed and stopped seems very similar to the effect of a head-on collision between the robot and a transparent or otherwise undetectable wall. In either case, it is important for the robot to be able to detect the situation and react accordingly.

(a) Collision

(b) Hold

(c) Stuck

Figure C.1: Types of Motion Interference considered in this work: (a) Collision against a partially detectable obstacle, (b) Being held by a person, and (c) Having one or more wheels stuck

**Having one or more wheels stuck (*stuck*)** Several events in the environment might cause one or more of the robot's wheels to get stuck. For example, a person might accidentally place their foot in front of the robot's wheels during navigation (see Figure C.1c), or the robot could have trouble passing over gaps or level changes, such as the entrance into an elevator. Additionally, a similar type of motion interference might occur independently of the environment due to malfunctioning wheel motors.

Note that all of the described forms of interference will have a similar effect on the robot: the robot's motion in the direction of travel will be impeded, and thus its velocity is going to be diminished to a value smaller than the given velocity command. Because of this similarity, all of these events are grouped for this work under the category of Motion Interference, and are treated as equivalent events. Experimental results in Section C.3 support this abstraction. In the case of non-equivalent events, however, the model can readily support detection of different types of events by adding the appropriate states to the model of Section C.2.

## C.2 A HMM for Motion Interference detection

Hidden Markov Models are particularly well-suited for modeling an *MI* detector: even though the CoBots don't have sensors to directly detect *MI* events, the occurrence of these events can be inferred from the observations that are available to the robot. HMMs provide an appropriate framework to perform these inferences.

A Hidden Markov Model $M$ can be defined as a 5-tuple $M = \{S, \Pi, A, O, B\}$, where

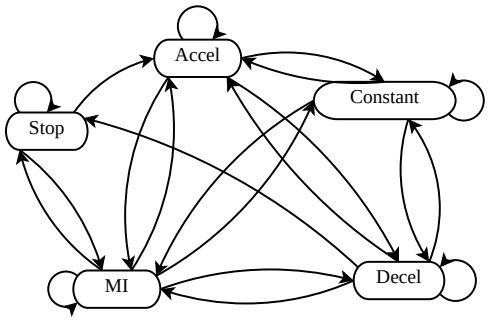



Figure C.2: Diagram of the HMM modeling the Motion Interference Monitor. Ellipses represent hidden states, while arrows represent transitions between states.

| | |
|---|---|
| $S = \{s_i\}$ | the set of hidden states in $M$ |
| $\Pi = \{\pi_i\}$ | the initial distribution of $M$ such that $P(S_1 = s_i) = \pi_i$ |
| $A = \{a_{ij}\}$ | the transition probabilities such that $P(S_{t+1} = s_j \mid S_t = s_i) = a_{ij}$ |
| $O = \{\vec{o}_i\}$ | the space of possible observations |
| $B = \{b_i\}$ | the emission probabilities such that $P(\vec{o}_j \mid S_t = s_i) = b_i(\vec{o_j})$ |

For the detector described in this work, each possible simple behavior of the robot is represented by a state in $S$. The possible behaviors assigned to the CoBot for purposes of *MI* detection are $S = \{stop, accel, constant, decel, mi\}$ representing the states where the robot is stopped, accelerating, at constant positive speed, decelerating, and having its motion interfered, respectively. A diagram for the resulting HMM is shown in Figure C.2. Since the robot always starts from the $stop$ state, the respective values for $\Pi$ are $\Pi = \{1, 0, 0, 0, 0\}$.

Transition probabilities between pairs of states were determined from hand-labeled gathered training data. In general, given a set of observations accompanied by state labels $S_t$ for all times $0 \leq t \leq T$,

$$a_{ij} = \frac{\sum_{t=0}^{T-1} \left(\delta_{S_t,s_i} \cdot \delta_{S_{t+1},s_j}\right)}{\sum_{t=0}^{T-1} \delta_{S_t,s_i}}, \tag{C.1}$$

where $\delta_{i,j}$ is the Kronecker delta function. That is, the transition probability from state $s_i$ to $s_j$ is given by the total number of labeled observations in which the robot transitioned from state $s_i$ to $s_j$ divided by the total number of labeled observations in which the robot transitioned from $s_i$ to anywhere. The only transition probabilities not calculated using Equation C.1 were transitions into *MI* $a_{i,mi} \forall i \neq mi$. Given the rarity of *MI* events in normal CoBot runs, *MI* events had to be artificially created to gather significant data for experimenting, and thus the real probabilities to transition from a different state to the *MI* state are extremely difficult to gather. Because of this, these values were set to $a_{i,mi} = p_{mi}$, where $p_{mi}$ is the only parameter for the *MI* detector. Tuning $p_{mi}$ has the effect of varying the total number of *MI* events detected, and therefore it serves as a parameter to find the desired trade-off value between precision and recall of *MI* events (see Section C.3 for more details).

Observation space $O$ may vary greatly depending on the sensors of the specific robot and the type of event that needs to be detected. For the task of detecting *MI* events in the CoBot, observations were of the form

$$\vec{o_t} = (u_t - v_t, a_t, j_t, u_t), \tag{C.2}$$

where $u_t$ is the commanded forward velocity of the robot, and $v_t$, $a_t$ and $j_t$ are the forward velocity, acceleration and jerk of the robot as measured by its wheel encoders. The reasoning for each observation and the method for obtaining it are the following:

**Velocity difference $\mathbf{u_t - v_t}$.**

One can expect that when the robot's movement is being interfered, its measured velocity would be significantly lower than its commanded velocity. While $u_t$ is directly obtained as a command, $v_t$ needs to be calculated from the individual encoder velocities. Velocity $v_t$ was obtained from the least squares solution transforming the encoder values for each of the four wheels to forward, sideways and rotational components of the robot's velocity.

**Acceleration $\mathbf{a_t}$.**

There are times during normal (i.e., not *MI*) robot motion when the velocity difference $u_t - v_t$ is significantly positive. For example, when the robot accelerates from a stopped position to full speed, the change in $u_t$ is discrete, while $v_t$ smoothly changes from 0 to $u_t$ over time. Thus, $u_t - v_t$ alone is not a sufficient observation to detect *MI* events. Therefore, acceleration $a_t$ is added as an additional layer to distinguish between these events: while an accelerating robot has a positive acceleration, a robot during a *MI* event usually has either negative (at the moment the *MI* event begins) or near-zero (once velocity has stabilized) acceleration. $a_t$ can be obtained by applying linear regression to the last $N_a$ measures of velocity $v_t$ as a function of $t$, and then getting the slope of the resulting line. Even though $a_t$ could be obtained from only the last 2 values of $v_t$ and $t$, a larger number $N_a$ is used to reduce the effects of noise in the data. $N_a$ must be large enough to negate the effects of noise, but small enough to provide a meaningfully recent value for $a_t$. For this work, $N_a = 4$.

**Jerk $\mathbf{j_t}$.**

In the event that the robot's motion is disrupted while the robot is accelerating, it might be the case that the positive acceleration continues for a while until the final velocity under the disturbance is reached. In these cases, $a_t$ might be within the normal parameters of an accelerating motion at any instant, but the change in acceleration in time may provide valuable information to distinguish *MI* acceleration from normal acceleration. For this reason, the second time derivative of the velocity is also used as part of the observations. Jerk $j_t$ is obtained by dividing the difference between the two last acceleration measurements by the difference in time between measurements. However, since jerk is significantly more sensitive to noise in the data than acceleration, a larger number of measurements $N_j \geq N_a$ is used to calculate the accelerations to be used for $j_t$, for purposes of further smoothing of noise at the cost of a more outdated jerk measurement. For this work, $N_j = 8$.

**Velocity command $\mathbf{u_t}$.**

The final attribute of the observation is simply the commanded velocity $u_t$. The only purpose of

this attribute is to distinguish between the $stop$ state, where $u_t = 0$, and the $constant$ state, where $u_t \neq 0$. Otherwise these states would have identical properties.

There are a couple of design decisions behind the use of the attributes of Equation C.2 as observations. First, notice that only the forward velocity, acceleration and jerk of the robot are used. While the CoBot's base is omnidirectional, most of its movements are restricted to the forward direction (plus rotations), given that most of its sensors are pointing forward. Therefore, using only the forward direction has the benefit of requiring estimation of fewer parameters at little cost. Furthermore while the CoBot has other sensors (e.g., Kinect, infrared sensors) that could provide estimates of velocity apart from the encoders, for this work only encoder estimates were used. While encoders provide the simplest method for velocity estimation, the biggest concern with using them as the only estimator is that, on extremely slippery surfaces, the robot's wheels could keep spinning at the commanded velocity even during an ***MI*** event. We determined, however, that even in the most slippery surfaces in the CoBots' environment (i.e., hardwood floors), slipping was not enough to make ***MI*** events undetectable using only encoder-based velocities. If this were not the case, however, additional sources of velocity information could be added as observations at the cost of additional parameter estimation.

Finally, emission probabilities $B$ are calculated from hand-labeled training data in a similar manner to transition probabilities $A$. For the specific monitor described in this work, two assumptions are made about the data: the attributes in the observation are conditionally independent, and the first three can be modeled as Gaussian distributions (the fourth attribute needs not be a Gaussian because all that matters is whether it is equal to 0 or not). While neither of these assumptions strictly hold for the case in question, they are good enough approximations, as evidenced by the results in Section C.3. Furthermore, these assumptions are not required in the model, but they simplify it; therefore, in cases where these assumptions are too strong for a detector to perform well, they can be eliminated at the cost of further model complexity and parameter estimation. Rewriting the observation attributes as $(o_1 \equiv u_t - v_t, o_2 \equiv a_t, o_3 \equiv j_t, o_4 \equiv u_t)$ for brevity, the emission probabilities can then be written as:

$$\begin{aligned} b_i(o_1, o_2, o_3, o_4) &= P(o_1, o_2, o_3, o_4 | S_t = s_i) \\ &= \prod_{j=1}^{4} P(o_j | S_t = s_i) \\ &= \prod_{j=1}^{3} f(o_j; \mu_{i,j}, \sigma_{i,j}^2) \cdot P_4(o_4, i), \end{aligned} \tag{C.3}$$

where the individual probabilities are defined as:

$$f(o_j; \mu_{i,j}, \sigma_{i,j}^2) = \frac{1}{\sigma_{i,j}\sqrt{2\pi}} e^{-\frac{(o_j - \mu_{i,j})^2}{2\sigma_{i,j}^2}} \tag{C.4}$$

$$P_i(o_4) = \begin{cases} \delta_{0,o_4} & \text{if } i = 1 \\ 1 & \text{if } i = 2, 4, 5 \\ 1 - \delta_{0,o_4} & \text{if } i = 3 \end{cases} \tag{C.5}$$

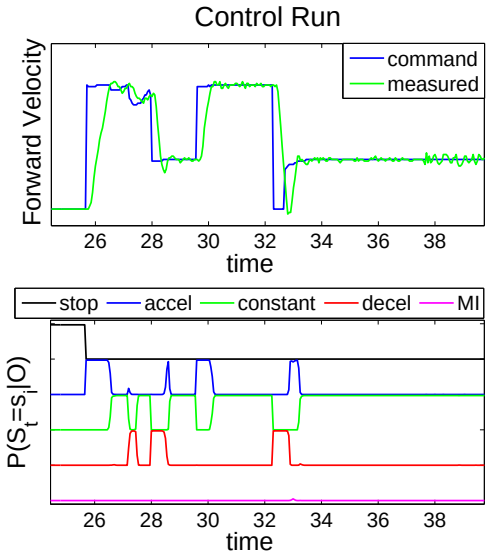



Figure C.3: Sample of data gathered from a normal (control) run of the CoBot navigating in its environment. The top figure shows the velocity command and the measured velocity over time, while the bottom stacked probability figure shows the respective assigned probabilities $P(S_t = s_i|O)$ (each between 0 and 1) for each state given the sequence of observations.

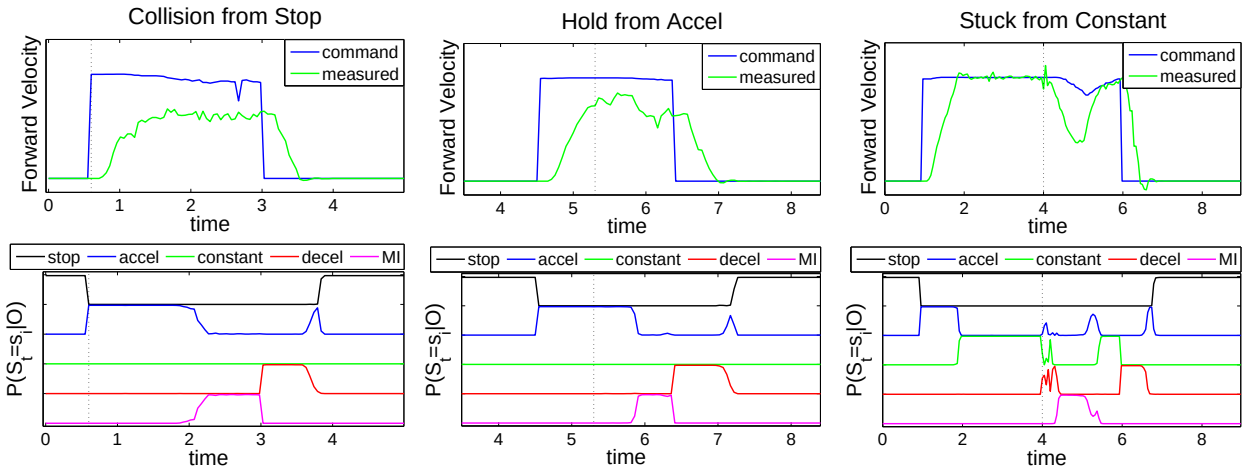
Equation C.4 simply describes a Gaussian probability distribution whose parameters were obtained from training labeled data, while Equation C.5 describes the probability of observing a certain velocity command $u_t = o_4$ depending on the current state: if the state is $stop$, then $o_4 = 0$ always; if the state is $constant$, then $o_4 \neq 0$, and in any other state $o_4$ could be anything.

Having defined all $S, \Pi, A, O, B$, the HMM-based *MI* detector is fully defined. Now, given any sequence of observations, the probability of being in state *MI* at each time can be calculated using an algorithm such as the forward algorithm described in [80]. Then, if $P(S_t = s_{mi}|O) > thresh$ (for this work, $thresh = 0.5$), an *MI* event has been detected at time $t$.

## C.3 Detector performance results

### C.3.1 Methods

To gather the necessary data for training and testing of the detector, two long control runs (no *MI* events) and 29 short test runs (with *MI* events) were conducted on the CoBot robot. For each run, the robot was instructed at a high level to move from its current location to a different location in the building; during navigation, the driving commands, encoder-based velocity and times of *MI* events (perceived by a human observer) were recorded. The control runs, during which a human supervisor made sure no *MI* events happened, lasted about 3 minutes each, and gave a total of 7145 observations. A subset of the data gathered during a control run is shown in Figure C.3. The test runs were each much shorter, focusing on the *MI* event, and giving a total of 8101 observations. Of the test runs, 15 contained $collision$ events, 6 contained $hold$ events, and 8 contained $stuck$




(a) Collision data (b) Hold data (c) Stuck data

Figure C.4: Examples of data gathered from *MI* runs. As in Figure C.3, top figures show velocities while bottom figures show probabilities for each state. Figures show (a) a collision against a partially detectable obstacle right as the robot starts to move, (b) somebody holding the robot as it is accelerating, and (c) something interfering with a wheel's rotation when the robot is traveling at constant speed. Vertical dotted lines indicate the beginning of an *MI* event, while rise in $P(S_t = s_{mi}|O)$ above 0.5 indicates detection of the event.

events. Figure C.4 shows the data gathered from three of these experiments.

To train and test the detector, for each run of the robot, each observation was manually labeled as one of $stop$, $accel$, $constant$, $decel$ or *MI*. Given that the *MI* times were previously recorded, each observation during those periods was labeled as *MI*. When the robot travels at constant speed (and similarly at 0 speed), the standard deviation of the velocity measured from the encoders is about $\sigma = 0.028$m/s. From this, each observation where the velocity command $u_t$ was 0 and the measured velocity $v_t$ was within $\sigma$ of 0 was labeled as $stop$. Similarly, each observation where $|v_t - u_t| \leq \sigma, u_t \neq 0$ was labeled as $constant$. Labeling $accel$ (or $decel$) observations consisted of the following: every observation after an increase (or decrease) of $u_t$, and while $|v_t - u_t| > \sigma$ was labeled as $accel$ (or $decel$, respectively). Other observations (i.e., noisy observations where $|v_t - u_t| > \sigma$ but $u_t$ had been constant) were labeled as $constant$.

The performance of the detector was then tested using leave-one-out cross-validation: for a given parameter set, 31 tests were conducted (one for each labeled run of the robot). For test $i$, all runs except for run $i$ were used for training the detector (i.e., obtaining the transition and emission probabilities of the HMM), which was then tested on run $i$. A *true positive* detection happened when at least one frame within a time interval labeled as *MI* was classified as *MI*. A *false positive* detection was defined as each group of consecutive frames outside of the time intervals labeled as *MI* that were classified as *MI*, given that such a group was not just a continuation of a true positive detection (sometimes the probability of being in state *MI* could take a few frames to decay after the robot had overcome the *MI* event; this was not considered a false positive). A *false negative* was defined as each *MI*-labeled time interval where no *MI* event was detected.

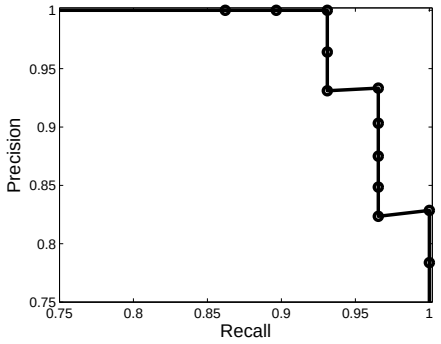



Figure C.5: Trade-off between precision and recall rates for motion interference detection, as parameter $p_{mi}$ is varied.

### C.3.2 Results

The goal of the *MI* detector presented in Section C.2 is to detect *MI* events reliably and within a useful time from initial interference. The proposed measures for judging the model are therefore precision and recall rates of *MI* event detection, as well as the average and median time to detection from when interference starts.

The only parameter of the model, transition probability $p_{mi}$, was varied to find the trade-off between precision and recall rates. For each tested parameter value, a *full test* –i.e., a set of 31 cross-validation tests as described in Section C.3.1– was conducted, giving the results shown in Figure C.5. For the purposes of the CoBot project, it was essential that the model be optimized for precision, given that the project focuses on giving a high degree of autonomy to the robot, and stopping execution for false positive detections would hinder this autonomy.

An optimal parameter value for our purposes was $p_{mi} = 5 \times 10^{-8}$, since it had the highest recall rate for which 0 false positives were detected. For this value, the precision rate was 100% while the recall rate was 93.1%; the average time to detection from initial motion interference was $\bar{t} = 0.647$ seconds. The median time to detection was significantly lower than this, at $t_m = 0.36$ seconds, reflecting the fact that a few outlier detections took significantly longer than the average detection time. These outliers were mostly *MI* events that started from the $stop$ state (e.g., Figure C.4a); this can be explained by the fact that the wheel's accelerations looked normal in the beginning, even if they were mostly slipping while spinning, before their behavior was abnormal enough to be detected as an *MI* event. This suggests that perhaps adding an estimate of velocity from a different sensor as an observation could help diminish the average time of detection. Overall, however, the detection time was usually well under a second, which is a useful time-frame for many applications, such as stopping when being held from a dangerous situation, or reacting to an inescapable collision.

## C.4 Discussion

We presented an HMM-based model for Motion Interference (*MI*) detection for a mobile robot. We identified the sensory observables of the robot and three types of motion interference. Through experiments conducted on the CoBot service robot, we have shown that such a model can successfully detect events that are not directly perceivable by the robot. Our work in general contributes an approach in which robots can reason about specific events by looking at their internal and external sensed state with input from their commanded controls. While this work focuses on the detection of *MI* events rather than actions to recover from them, we considered a base stop command to the robot when the event is detected, and will pursue research on other possible actions.

The *MI* events we considered limit the forward velocity of the robot, but other types of *MI* events could be detected using a similar approach (e.g., pushing the robot so that its measured velocity is above its velocity command). It is in principle feasible to detect anomalies in the behavior of the robot even if these anomalies have not been explicitly modeled: the formulation of HMMs allows us not only to calculate the probability of being in a particular state given a series of observations, but also the probability that a model describes the observable of a robot given a particular series of observations [80]; one could thus expect that a robot that has fallen in an unmodeled state would yield a significantly different model probability distribution than a robot running normally (i.e., within the model). In this way, HMM-based models for execution monitoring could provide a natural model for implementation of hybrid model-based and model-free monitoring. Finding whether this is a practical method to detect anomalies in our robots is a topic of future research.