Detecting and Correcting Model Anomalies in Subspaces of Robot Planning Domains

Juan Pablo Mendoza
The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, 15213
jpmendoza@ri.cmu.edu

Manuela Veloso
Computer Science Department
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, 15213
mmv@cs.cmu.edu

Reid Simmons
The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, 15213
reids@cs.cmu.edu

ABSTRACT

Making decisions based on accurate models enables robots to exploit domain knowledge to act intelligently. However, in many realistic domains, it is impossible to have globally accurate models, as the world may exhibit modes of behavior during deployment that were unforeseeable during model building. This paper addresses the problem of adaptation in domains in which robots have access to a model of the world that is generally accurate, but which is inadequate in particular sets of similar situations –i.e., subspaces of the task domain. Using optimization techniques to find parametric approximations to these subspaces, our framework generalizes from sparse observations to find and correct for statistical incongruences between expected and observed behavior. We demonstrate this framework in a domain in which single deployment adaptation is essential: a team of soccer robots keeping the ball away from a previously unknown opponent. Empirical results show that the framework improves model accuracy and task performance over timescales comparable to a single soccer game.

Categories and Subject Descriptors

I.2.9 [**Robotics**]

Keywords

Multi-robot systems; Robot planning and plan execution; Single and multi-agent learning; Fault tolerance and resilience.

1. INTRODUCTION

Robots often use models of the effects of their actions to make intelligent decisions. Unfortunately, in most realistic environments, it is infeasible to have the perfect knowledge and computational resources required to create globally accurate models. In particular, several robotics applications (e.g., space robots and robots in adversarial domains) require deployment in environments where thorough previous testing is infeasible. Thus, while a generally accurate initial model may be available, there may be situations in which the world exhibits different behavior from that predicted by the initial model.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

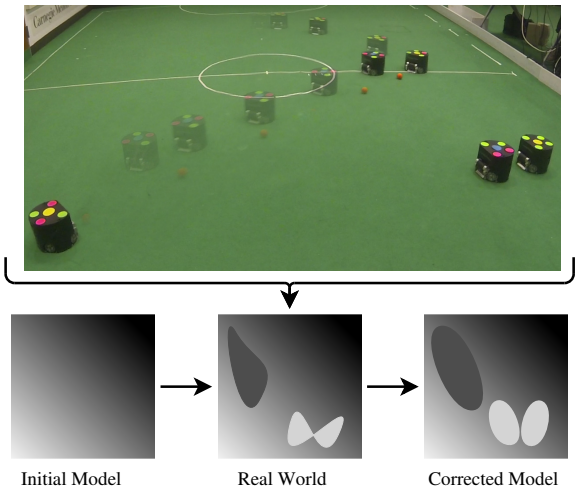


Figure 1: Robots with a generally accurate initial model of their domain may experience during execution that their model is inadequate in some specific subspaces. Our framework finds these subspaces and creates a corrected model that approximates them parametrically.

In this work, we address the problem of model correction for domains in which these deviations from nominal behavior occur in particular sets of similar situations, represented as sets of *subspaces of a state-action feature space* of the robots. The robot does not know before deployment whether such anomalous subspaces exist, or their shape in such feature space, and must therefore find the extent of these anomalies and correct its models accordingly at execution time.

Furthermore, we are interested in domains in which performance over a single robot deployment may be crucial. Our approach thus needs to correct models online during execution, and generalize from the potentially sparse observations that happen during one deployment. With these goals in mind, we present an approach that incrementally finds parametric approximations to anomalous subspaces as the robot gathers new observations, and corrects the model accordingly. A sketch of this process is shown in Figure 1.

As a motivating domain, and for empirical demonstration, we apply our framework to a modified version of the Keepaway problem [18] in which a team of robots must keep the ball away from opponent robots by passing it to each other, as shown in Figure 1. In particular, we focus on the deci-

sion that the robot currently holding the ball must make: it needs to choose where to pass next such that its team will remain in possession of the ball. The Keepaway problem has the key properties that we wish our framework to address:

Generally accurate model: Our robots have accurate models of the motion of the ball and of their own ball interception capabilities. The opponent's motion profiles can be easily measured, so it is possible to create a rough model of how they can intercept the ball (e.g., assume they use the same algorithms as our robots).

Anomalous subspaces: While our robots have a general model of how opponents intercept the ball, each opponent the team encounters is likely to demonstrate various unexpected skills in particular situations, or to behave suboptimally in others (according to our definition of optimality). We want our robots to detect such deviations and adapt to maximize performance.

Performance in a single deployment: In robot soccer games, robots often play a single game against each previously unknown opponent. Therefore, in this domain, as in many other prominent ones (e.g., space robots, hostile environment robots), good performance in a single deployment is crucial.

The key idea of our work is that single deployment adaptation in such domains can be achieved by (1) searching for the subspaces of the domain in which observations deviate significantly from the model, and then (2) correcting the planning model in those subspaces using the observations experienced in them. Using a framework that integrates anomalous subspace detection as a means to monitor execution of plans [13], this paper introduces the following contributions: (1) a mathematical framework and algorithm for detection of multiple anomalous subspaces, (2) a mathematical framework and algorithm for planning model correction based on such detections, and (3) empirical evidence demonstrating that this method can enable improvement in model prediction and task performance in a realistic robot deployment.

2. RELATED WORK

Our work is closely related to previous work on robot *execution monitoring* [15]: the problem of detecting failures during execution, diagnosing the reasons for these failures, and recovering from them. Expectation-based execution monitoring [4, 6] is particularly relevant to our work. Expectation-based monitors detect failures in execution by comparing *expectations* generated by the planner to observations received during execution. While we are also interested in comparing planner-generated expectations to execution-generated observations to detect anomalies, our work differs in that it focuses on monitoring stochastic expectations and finding subtle anomalies from collections of observations, rather than detecting failure from a single observation. Furthermore, we focus less on diagnosing the reason for failure, and more on adapting the model to improve performance based on experience.

Model adaptation is an essential component of the problem we address. Several Reinforcement Learning (RL) algorithms have addressed the problem of learning to perform well in continuous environments that are not perfectly modeled. Model-free RL approaches, such as Q-Learning [5] and policy gradient descent [19], have been shown to achieve great performance in various robotics domains without explicitly modeling the world. In fact, the Keepaway soccer problem was first analyzed in the context of model-free RL [18]. 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 [1]. Unlike model-free approaches to adaptation, we seek to exploit domain knowledge to allow our robots to focus their learning toward modifying their models in particular anomalous subspaces, using few observations.

Model-based RL approaches learn, along with optimal policies, a transition model for the Markov Decision Process that describes their world [8]. The work presented here is indeed an instance of Model-based RL: our robots update their model of the world based on experience, and then create plans based on the improved model. In this paper, we focus on the problem of updating these models in domains in which particular subspaces of the world are not well-modeled initially. While previous work has addressed the problem of learning situational-dependent models [7], our work focuses on online learning by generalizing over potentially large subspaces from sparse data, rather than building precise models from abundant data.

Since our approach relies on the search for anomalous subspaces of the domain, work in the field of anomaly detection is closely related. Much of the work on anomaly detection has focused on detecting single outlier observations [3], while we are interested in finding anomalies in the distribution of collections of observations. In that sense, our work is related to time-series anomaly detection [10, 9], but we focus on data that is spatially related in some state-action feature space, rather than in purely temporal relationships. Previous work has addressed the problem of detecting anomalies in spatially related data [11, 14], and indeed some of the statistics used in our work were inspired by this related work. However, the algorithms used to find such anomalies were designed for two or three-dimensional domains, and do not scale well to real-time monitoring in higher dimensions. Furthermore, the related spatial anomaly detection work was not conducted in the field of robotics, making detection their ultimate goal, rather than model correction for task performance improvement.

3. EXPECTATION-BASED MONITOR

The goal of this work is to improve robot performance by incorporating information acquired during execution back into the planner model. To do this, we use an expectation-based monitor [13], briefly described here for completeness. In this monitor, the planner generates *expectations* about the possibly non-immediate effects of taking an action $a \in A$ in state $\boldsymbol{s} \in S$. These expectations are predictions about the distribution of a random variable $\boldsymbol{z}$ whose outcome can be *observed* during execution.

In the monitor for the Keepaway domain, given the state $\boldsymbol{s}$ of the robots and ball, the planner decides where to pass the ball (action a) based on its *expectation* about the probability of success $P(\boldsymbol{z} = 1|\boldsymbol{s}, a) = P(\boldsymbol{z}|\boldsymbol{\theta}(\boldsymbol{s}, a))$ of each pass, as determined by parameters $\boldsymbol{\theta}(\boldsymbol{s}, a)$.

During execution, the outcome of a pass can only be *observed* once the world enters a state $\boldsymbol{s_z} \in S_{\boldsymbol{z}}$ in which the pass has ended. At this point, the pass is observed to be a

success ($\boldsymbol{z} = 1$) if one of our robots has the ball, or a failure ($\boldsymbol{z} = 0$) if an opponent robot has the ball.

In general, then, the planner needs to give the execution monitor expectations e consisting of the state $\boldsymbol{s}$ of the world when the action was taken, the action a taken, and a set of pairs containing the states $S_{\boldsymbol{z}}$ in which the outcome of a can be observed, and the expected distribution parameters $\boldsymbol{\theta}$ of such observation: $\mathcal{P} = \{\boldsymbol{s_z}, \boldsymbol{\theta}(\boldsymbol{s}, a, \boldsymbol{s_z}) | \boldsymbol{s_z} \in S_{\boldsymbol{z}}\}$ An expectation is thus defined as:

$$e(\boldsymbol{s}, a) = (\boldsymbol{s} \in S, a \in A, \mathcal{P} \subseteq (S \times \boldsymbol{\Theta}))\,. \tag{1}$$

In our particular domain, $P(\boldsymbol{z}|\boldsymbol{\theta}(\boldsymbol{s}, a, \boldsymbol{s_z})) = P(\boldsymbol{z}|\boldsymbol{\theta}(\boldsymbol{s}, a)) = P(\boldsymbol{z} = 1|\boldsymbol{s}, a)$. However, more generally, $\boldsymbol{\theta}$ could be a function of the evaluation state $\boldsymbol{s_z} \in S_{\boldsymbol{z}}$ (c.f. [13]).

Given that the planner generates a list of expectations E, Algorithm 1 describes the monitoring framework followed in this paper, which runs every time step of execution.

Algorithm 1 Execution monitor procedure run every time step t of execution, given the current world state $\boldsymbol{s}_t$ and the list of pending expectations E generated by the planner

```
1: function MONITOR(s_t, E)
2:     for each e = (s, a, P) ∈ E do
3:         if ∃ (s_z, θ) ∈ P s.t. s_z = s_t then
4:             z ← Observe(s, a, s_t)
5:             z' ← (x(s, a), z, θ)
6:             add z' to observations Z.
7:             remove e from E
8:         end if
9:     end for
10:
11:    R ← DMAPS(Z)        ▷ Find anomalies, Algorithm 2
12:    UpdateModel(R)                    ▷ Equation 23
13: end function
```

The monitor first determines whether there are pending expectations that can be verified in the current state $\boldsymbol{s}_t$: For every expectation in E, the monitor checks whether $\boldsymbol{s}_t$ is an element of the expectation evaluation states in $\mathcal{P}$ (line 3). If so, then an observation is generated through the domain-specific function Observe. In the Keepaway domain, when the the pass ends, function Observe$(\boldsymbol{s}, a, \boldsymbol{s_z})$ returns 1 if the pass was successful, and 0 otherwise[1].

Once an observation is generated, an augmented observation $\boldsymbol{z}' = (\boldsymbol{x}(\boldsymbol{s}, a), \boldsymbol{z}, \boldsymbol{\theta})$ is created from it, consisting of a feature vector $\boldsymbol{x} \in \mathcal{X}$ of the state and action which led to $\boldsymbol{z}$, the observation $\boldsymbol{z}$, and the expected distribution parameters $\boldsymbol{\theta}$ of $\boldsymbol{z}$. The list of all such augmented observations is the input given to the anomalous subspace detector (line 11), which does the core work of the execution monitoring by finding anomalous subspaces of the domain as described in Section 4. Finally, the planning model is corrected based on information from the detected anomalous subspaces $\mathcal{R}$, as described in Section 5.

In practice, lines 11 and 12 run on a separate thread from the rest of execution and monitoring, because the control loop of the robots needs to run at 60 Hz, while finding anomalous subspaces can take over a second. This means that any time the monitor runs, it uses the latest execution data available at the time it begins. Similarly, the planner uses the latest monitoring corrections available at the time it begins the planning process.

[1]While the observables in the domain presented here are Bernoulli variables, we accumulate these observations into discrete bins in state-action space for higher computational efficiency. These observations thus follow a Binomial distribution. For ease and generality of implementation, we use the normal approximation to binomial distributions.

4. ANOMALOUS SUBSPACE DETECTION

To detect multiple anomalous subspaces in which execution does not match expectations, we extend the FARO algorithm [12], formulated for domains with at most one anomalous subspace.

FARO can be used to find a subspace R of the state-action feature space $\mathcal{X}$ in which measured observations deviate in a statistically significant way from the expected distribution of such observations. To do this, FARO uses standard nonlinear optimization algorithms to find the parametric subspace $R_{\max} \subseteq \mathcal{X}$ that maximizes an anomaly value $\text{anom}(R, \boldsymbol{Z})$ of the observations $\boldsymbol{Z}$, where $\text{anom}(R, \boldsymbol{Z})$ is a likelihood ratio:

$$\text{anom}(R, \boldsymbol{Z}) = \frac{P(\boldsymbol{Z}|R \text{ is anomalous})}{P(\boldsymbol{Z}|R \text{ is normal})} \tag{2}$$

More specifically, we assume that (a) observations are independent of each other given the state-action that generates each of them, and (b) observations generated by anomalous execution are distributed according to parameters taken from a (potentially infinite) set of parameter functions $\hat{\boldsymbol{\Theta}}$:

$$\text{anom}(R, \boldsymbol{Z}) = \max_{\hat{\boldsymbol{\theta}} \in \hat{\boldsymbol{\Theta}}} \left[\prod_{(\boldsymbol{x}_i, \boldsymbol{z}_i, \boldsymbol{\theta}_i) \in \boldsymbol{Z}} \mathbb{1}(\boldsymbol{x}_i \in R) \frac{P(\boldsymbol{z}_i|\hat{\boldsymbol{\theta}}(\boldsymbol{x}_i))}{P(\boldsymbol{z}_i|\boldsymbol{\theta}_i)} \right] \tag{3}$$

In this work, $\hat{\boldsymbol{\Theta}}$ is the set of distributions that shift the mean of the nominal distribution by some vector $\boldsymbol{\delta}$ unknown a priori[2]. For brevity, since $\boldsymbol{Z}$ is constant throughout each step of optimization, we often omit it from the list of function arguments (e.g., we write $\text{anom}(R, \boldsymbol{Z})$ as $\text{anom}(R)$).

The formulation of Equation 3 does not trivially generalize to domains with multiple anomalies. Section 4.1 addresses this issue, describing why the generalization is not trivial, and proposing an alternate formulation of the problem which includes multiple anomalous subspaces. Section 4.2 grounds this formulation into an algorithm for Detection of Multiple Anomalous Parametric Subspaces (DMAPS).

4.1 Detection of multiple anomalies

A natural (but ultimately insufficient) first idea on how to extend the formulation of Equation 2 to domains with multiple anomalies would be to simply maximize an analogous function $\text{Anom}'(\mathcal{R})$ over *sets $\mathcal{R}$ of subspaces*:

$$\text{Anom}'(\mathcal{R}) \equiv \prod_{R \in \mathcal{R}} \frac{P(\boldsymbol{Z}|R \text{ is anomalous})}{P(\boldsymbol{Z}|R \text{ is normal})} \tag{4}$$

This formulation is inadequate for two reasons: (1) overlap among subspaces 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

[2]The relevance of this case is described in previous work [12]; the ideas generalize to other sets of potential anomalies.

anomalous subspaces, each with one observation, would be considered as anomalous as a simpler hypothesis with one anomalous subspace with those n observations.

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

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 modeled behavior, or one of the unmodeled modes of the world.*

In practice, what this assumption implies is that a single observation cannot contribute to the anomaly value $\text{anom}(R)$ of more than one subspace 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 subspaces that that maximizes $\text{Anom}'(\mathcal{R})$, we search for the set $\mathcal{R}_{\max}$ that maximizes the following function $\text{Anom}(\mathcal{R})$:

$$\text{Anom}(\mathcal{R}) \equiv \frac{P\left(\forall R \in \mathcal{R}.\ \ R \text{ is anomalous}|\boldsymbol{Z}\right)}{P\left(\forall R \in \mathcal{R}.\ \ R \text{ is normal}|\boldsymbol{Z}\right)}. \tag{5}$$

Furthermore, we state a second assumption on the problem:

ASSUMPTION 2. *For any two subspaces $R_i \subseteq \mathcal{X}$ and $R_j \subseteq \mathcal{X}$, "$R_i$ is an anomalous subspace" is conditionally independent from "R_j is an anomalous subspace", given the list of observations $\boldsymbol{Z}$.*

In domains in which $\boldsymbol{Z}$ is the only source of information about the presence of anomalous subspaces, this assumption holds naturally. However, it is possible to think of 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 subspaces. Addressing these domains is beyond the scope of this paper.

The proposed formulation of Equation 5, along with Assumption 2, lead to a more desirable optimization cost function, which naturally favors simpler hypotheses, as shown by Theorem 1 below:

THEOREM 1. *Given Assumption 2, the set $\mathcal{R}_{\max}$ of subspaces that maximizes* $\text{Anom}(\mathcal{R})$ *is also the set that maximizes the simpler function*

$$F(\mathcal{R}) \equiv \left[\sum_{R\in\mathcal{R}} \log(\text{anom}(R))\right] - \sum_{R\in\mathcal{R}} \lambda(R), \tag{6}$$

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

PROOF. Given the assumption above, the probabilities of two different regions being anomalous, given the data, are independent of each other. We can therefore rewrite Equation 5 as

$$\begin{aligned}
\text{Anom}(\mathcal{R}) &= \prod_{R\in\mathcal{R}} \frac{P(R \text{ is anomalous}|\boldsymbol{Z})}{P(R \text{ is normal}|\boldsymbol{Z})} \\
&= \prod_{R\in\mathcal{R}} \frac{P(\boldsymbol{Z}|R \text{ is anomalous})P(R \text{ is anomalous})}{P(\boldsymbol{Z}|R \text{ is normal})P(R \text{ is normal})} \\
&= \left[\prod_{R\in\mathcal{R}} \text{anom}\left(R\right)\right]\left[\prod_{R\in\mathcal{R}} e^{-\lambda(R)}\right].
\end{aligned} \tag{7}$$

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

$$\begin{aligned}
\arg\max_{\mathcal{R}\subseteq 2^{\mathcal{X}}} \left[\text{Anom}(\mathcal{R})\right] &= \arg\max_{\mathcal{R}\subseteq 2^{\mathcal{X}}} \left[\log(\text{Anom}(\mathcal{R}))\right] \\
&= \arg\max_{\mathcal{R}\subseteq 2^{\mathcal{X}}} \left[\left[\sum_{R\in\mathcal{R}} \log(\text{anom}(R))\right] - \sum_{R\in\mathcal{R}} \lambda(R)\right] \\
&= \arg\max_{\mathcal{R}\subseteq 2^{\mathcal{X}}} \left[F(\mathcal{R})\right]
\end{aligned} \tag{8}$$

□

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

$$F(\mathcal{R}) = \left[\sum_{R\in\mathcal{R}} \log(\text{anom}(R))\right] - |\mathcal{R}|\lambda, \tag{9}$$

where λ is a constant.

For the work in this paper, we assume λ to be a constant for all regions, although incorporating a non-uniform informative prior is a subject of interest for future work.

Notice that, in domains in which anomalies are less common than nominal behavior (i.e., $\lambda(R) > 0$), the cost function $F(\mathcal{R})$ naturally favors simpler hypotheses with fewer anomalous subspaces.

4.2 Anomalous Subspace Detection Algorithm

Optimizing function $F(\mathcal{R})$ is a hard, non-convex problem. Even the simpler problem of globally optimizing for a single region is not tractable [12], which is why FARO uses an iterative optimization approach. Here, we also use an iterative local optimization approach for each candidate subspace. Furthermore, our algorithm takes a greedy approach to optimize for multiple anomalies: anomalous subspaces are optimized in sequence, in non-ascending order of their value $\text{anom}(R)$ prior to the call.

Algorithm 2 describes the sequential optimization process of DMAPS. First, the algorithm adds a small candidate subspace around the most recent observation to $\mathcal{R}$ in line 3, similarly to the FARO algorithm. This set of subspaces is then sorted in non-increasing order of value $\text{anom}(R)$ for sequential optimization. At this point, in line 8, the sequential optimization over candidate subspaces begins.

For the optimization of each subspace R in line 10, we use the Cross Entropy Method (CEM) for randomized optimization [16]. After optimizing a subspace from R into R', the decision of whether to add this region to the final output set is made in line 11. A subspace 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 $\mathcal{R}_{\max}$ found to be most likely to be the set of unmodeled subspaces 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 subspace $R \in \mathcal{R}$ 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 counterexamples to its optimality can be found. In spite of this, the detection of DMAPS, combined with the correction presented in Section 5 has shown to significantly improve performance and prediction in realistic environments, as shown in Section 7.

Algorithm 2 DMAPS anomaly detector. Receives as input a list of observations $\boldsymbol{Z}$, and returns a set $\mathcal{R}$ of anomalous regions found, along with their corresponding anomaly values.

```
1: function DMAPS( Z = [z'_i | i = 0, ..., t])
2:     R ← Previous most anomalous regions
3:     R ← R ∪ ball(x_t)                    ▷ Ball around last obs.
4:     ℝ ← Sort(R)                          ▷ Descending order of anom(R)
5:
6:     R ← ∅                                ▷ Reset R and F(R)
7:     F ← 0
8:     for R ∈ ℝ do
9:         Optimize R into R' such that
10:                anom(R', Z) ≥ anom(R, Z)
11:        if log(anom(R', Z)) ≥ λ(R') then
12:            R ← R ∪ R'
13:            F ← F + log(anom(R', Z)) − λ(R')
14:        end if
15:    end for
16:    return R
17: end function
```

5. ANOMALOUS SUBSPACE MODEL CORRECTION

Once the DMAPS detector has found a set of regions $\mathcal{R}$ likely to be unmodeled subspaces of the world, this information can be used to correct the planner's world model. This corresponds to line 12 in Algorithm 1. We thus seek to compute a corrected model based on the nominal model[3] given by $\boldsymbol{\theta}^0$ and the set $\mathcal{R}$ of detected anomalous subspaces:

$$P(\boldsymbol{z}|\boldsymbol{\theta}^+) \equiv P(\boldsymbol{z}|\boldsymbol{s}, a, \boldsymbol{\theta}^0, \mathcal{R}) \tag{10}$$

We assume that, when the robot performs action a in state s, the world behaves according to its nominal behavior b_0 or one of the $|\mathcal{R}|$ behaviors b_R detected by DMAPS. We denote the set of plausible behaviors as $B = \{b_0\} \cup \{b_R | R \in \mathcal{R}\}$. By the law of total probability, we obtain:

$$P(\boldsymbol{z}|\boldsymbol{\theta}^+) = \sum_{b_i \in B} \left[P(\boldsymbol{z}|b_i, \boldsymbol{s}, a, \boldsymbol{\theta}^0, \mathcal{R}) P(b_i|\boldsymbol{s}, a, \boldsymbol{\theta}^0, \mathcal{R}) \right]. \tag{11}$$

It is clear then that the distribution of observations is a mixture of the distributions produced by the different plausible behaviors. We get a better idea of the parameters involved in Equation 11 by defining $\alpha_b(\boldsymbol{s}, a, \mathcal{R}) \equiv P(b|\boldsymbol{s}, a, \mathcal{R})$ and $P(\boldsymbol{z}|\boldsymbol{\theta}^b(\boldsymbol{s}, a)) \equiv P(\boldsymbol{z}|b, \boldsymbol{s}, a)$:

$$P(\boldsymbol{z}|\boldsymbol{\theta}^+) = \sum_{b \in B} \left[\alpha_b P(\boldsymbol{z}|\boldsymbol{\theta}^b) \right] \tag{12}$$

Section 5.1 discusses the problem of estimating $\boldsymbol{\theta}^b$, while Section 5.2 discusses the estimation of α_b.

5.1 Estimating observation distributions

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

[3]For simplicity of explanation we drop the potential dependence of $\boldsymbol{\theta}^0$ on the measurement state $\boldsymbol{s_z}$; the theory and algorithms apply equivalently in either case.

For the unmodeled behavior modes in each subspace $R \in \mathcal{R}$, we assume for this paper 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 4:

$$\boldsymbol{\theta}^R \equiv \arg\max_{\hat{\boldsymbol{\theta}} \in \hat{\boldsymbol{\Theta}}} \left[\prod_{(\boldsymbol{x}_i, \boldsymbol{z}_i, \boldsymbol{\theta}_i) \in \boldsymbol{Z}} \mathbb{1}(\boldsymbol{x}_i \in R) \frac{P(\boldsymbol{z}_i|\hat{\boldsymbol{\theta}}(\boldsymbol{x}_i))}{P(\boldsymbol{z}_i|\boldsymbol{\theta}_i)} \right] \tag{13}$$

In particular, for the examples of this paper, we assume Gaussian distributions and anomalies that shift the mean $\boldsymbol{\mu}$ of the distribution. In that case, the maximum likelihood distribution for behavior b_R is given by

$$P\left(\boldsymbol{z}|\boldsymbol{\theta}^R(\boldsymbol{s}, a)\right) = \mathcal{N}\left(\boldsymbol{\mu}_0(\boldsymbol{s}, a) + \boldsymbol{\delta}_R, \boldsymbol{\Sigma}_0(\boldsymbol{s}, a)\right). \tag{14}$$

Here, $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}_0$ are the parameters predicted by the nominal model, and $\boldsymbol{\delta}_R$ is the maximum likelihood shift in mean:

$$\boldsymbol{\delta}_R \equiv \left(\sum_{\boldsymbol{x} \in R} \boldsymbol{\Sigma}_i^{-1} \right)^{-1} \left(\sum_{\boldsymbol{x} \in R} \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{z}_i - \boldsymbol{\mu}_i) \right), \tag{15}$$

where $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the predicted mean and covariance, respectively, of the distribution of $\boldsymbol{z}_i$.

5.2 Estimating active behavior distributions

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

$$P\left(b_0|\boldsymbol{s}, a, \mathcal{R}\right) = 1 - \sum_{R \in \mathcal{R}} P\left(b_R|\boldsymbol{s}, a, \mathcal{R}\right), \tag{16}$$

so we focus on computing the probability of other behaviors.

The activation probability of each unmodeled behavior b_R is given by:

$$P(b_R|\boldsymbol{s}, a, \mathcal{R}) = P\left(b_R|\boldsymbol{x}(\boldsymbol{s}, a) \in R\right) P\left(\boldsymbol{x}(\boldsymbol{s}, a) \in R\right). \tag{17}$$

That is, the activation of behavior mode b_R depends both on whether feature state $\boldsymbol{x}(\boldsymbol{s}, a) \in \mathcal{X}$ lies inside of subspace R defining b_R, and on our confidence that R is actually an anomalous subspace with unmodeled behavior b_R.

For the work in this paper, $P\left(\boldsymbol{x}(\boldsymbol{s}, a) \in R\right)$ simply indicates whether a state-action point belongs to subspace R:

$$P\left(\boldsymbol{x}(\boldsymbol{s}, a) \in R\right) = \mathbb{1}(\boldsymbol{x}(\boldsymbol{s}, a) \in R). \tag{18}$$

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

For the confidence that R is actually an anomalous subspace with behavior b_R, it can be shown that

$$P\left(b_R|\boldsymbol{x}(\boldsymbol{s}, a) \in R, \boldsymbol{Z}\right) = P\left(R \text{ is anomalous}|\boldsymbol{Z}\right) \tag{19}$$

Notice that $P\left(R \text{ is anomalous}|\boldsymbol{Z}\right)$ is a monotonically increasing function of $\text{anom}(R)$, as defined in Equation 3. There exists a function $\mathbb{P} : \mathbb{R} \to \mathbb{R}$ such that $\mathbb{P}(\text{anom}(R)) = P\left(R \text{ is anomalous}|\boldsymbol{Z}\right)$. 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 $P(\text{anom}(R) > \gamma | R \text{ is nominal})$ [11][4]. Therefore, we estimate the desired confidence by:

$$P\left(b_R | \boldsymbol{x}(\boldsymbol{s}, a) \in R, \boldsymbol{Z}\right) = \hat{\mathbb{P}}(\text{anom}(R)) \tag{20}$$

Coefficient α_R is thus computed as:

$$\alpha_R(\boldsymbol{s}, a, \mathcal{R}) = \mathbb{1}\left(\boldsymbol{x}(\boldsymbol{s}, a) \in R\right) \hat{\mathbb{P}}(\text{anom}(R)) \tag{21}$$

5.3 Applying model corrections

Defining $\boldsymbol{\theta}^b$ and α_b for each plausible behavior fully defines Equation 12. Using normal distributions as described in Section 5.1 and α_b as defined in Section 5.2, we get:

$$P(\boldsymbol{z}|\boldsymbol{\theta}^+) = P(\boldsymbol{z}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)\left(1 - \sum_{R \in \mathcal{R}}\left[\mathbb{1}\left(\boldsymbol{x}(\boldsymbol{s}, a) \in R\right) \hat{\mathbb{P}}(\text{anom}(R))\right]\right) + \sum_{R_i \in \mathcal{R}}\left[P\left(\boldsymbol{z}|\boldsymbol{\mu}_0 + \boldsymbol{\delta}_R, \boldsymbol{\Sigma}_0\right) \mathbb{1}\left(\boldsymbol{x}(\boldsymbol{s}, a) \in R\right) \hat{\mathbb{P}}(\text{anom}(R))\right] \tag{22}$$

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

$$P(\boldsymbol{z}|\boldsymbol{\theta}^+) = \begin{cases} P(\boldsymbol{z}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)(1 - \hat{\mathbb{P}}(\text{anom}(R))) + & \text{if } \exists R \in \mathcal{R} \text{ s.t.} \\ P(\boldsymbol{z}|\boldsymbol{\mu}_0 + \boldsymbol{\delta}_R, \boldsymbol{\Sigma}_0)\hat{\mathbb{P}}(\text{anom}(R)) & \boldsymbol{x}(\boldsymbol{s}, a) \in R \\ P(\boldsymbol{z}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) & \text{otherwise} \end{cases} \tag{23}$$

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

$$\mathbb{E}[\boldsymbol{z}|\mathcal{R}] = \begin{cases} \boldsymbol{\mu}_0 + \boldsymbol{\delta}_R \hat{\mathbb{P}}(\text{anom}(R)) & \text{if } \exists R \in \mathcal{R} \text{ s.t. } \boldsymbol{x}(\boldsymbol{s}, a) \in R \\ \boldsymbol{\mu}_0 & \text{otherwise} \end{cases} \tag{24}$$

6. INTERCEPTION-KEEPAWAY DOMAIN

To demonstrate the framework presented here, we apply it to a realistic robot sub-problem of robot soccer. The domain, which we denominate Interception-Keepaway, is strongly inspired by the Keepaway domain introduced for 2D robot soccer simulation [18].

6.1 Interception Keepaway domain definition

The keepaway domain is a sub-problem of autonomous robot soccer that has become a benchmark problem in the 2D simulated robot league [17]. 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 [17]. 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 paper, 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 paper, 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.

6.2 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 ϕ_i of each robot, their first time derivatives $\boldsymbol{v}_b$, and the ball's 2D location $\boldsymbol{l}_b$ and velocity[5]. 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.

To employ the monitor described in this paper, we define an expectation-generating planner, as described in Section 3. We monitor the expected probability of success of passes, $P(\boldsymbol{z}|\boldsymbol{s}, a) = \boldsymbol{\theta}(\boldsymbol{s}, a)$. These expectations are 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

[4]We do not have access to the exact distribution of $\boldsymbol{Z}$ before execution, since the distribution is affected by model corrections made online. Here, we approximate $\hat{\mathbb{P}}$ from the distribution of the undisturbed model. In domains with higher computing power or softer efficiency constraints, $\hat{\mathbb{P}}$ could be computed online each time a new set $\mathcal{R}$ is discovered.

[5]For this paper, 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.

$\boldsymbol{v}_b$. Given such an estimate (e.g., [2]), the model computes the keepers' time to ball t_k:

$$t_k(\boldsymbol{l}_b, \boldsymbol{v}_b) = \min_{i \in \text{keepers}} t_i(\boldsymbol{l}_i, \boldsymbol{v}_i, \boldsymbol{l}_b, \boldsymbol{v}_b), \tag{25}$$

and similarly t_t for takers. The predicted probability of success is given by:

$$P(\boldsymbol{z}|\boldsymbol{s}, a = \boldsymbol{v}_b) = \Phi\left(\frac{t_t(\boldsymbol{l}_b, \boldsymbol{v}_b) - t_k(\boldsymbol{l}_b, \boldsymbol{v}_b)}{\sigma_t}\right), \tag{26}$$

Where Φ denotes the standard normal cumulative distribution, and σ_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.

Having defined the robot's prediction model, we now define the set of expectations E that get passed to the monitor of Algorithm 1. For each pass, m reasons e_i are created: one predicting the interception probability of each taker. These reasons can then be evaluated by the monitor in the following conditions ($\boldsymbol{s_z} \in \mathcal{P}$): If the pass is successful, then a success observation $\boldsymbol{z} = 1$ is generated for each of the m reasons; If the pass is intercepted by taker i, then a single failure observation $\boldsymbol{z} = 0$ is generated for reason e_i, and a void, informationless observation is generated for the rest of the reasons, as it is unknown whether any other taker would have been able to intercept the pass. To complete the definition of prediction set $\mathcal{P}$, we note that $\boldsymbol{\theta}(\boldsymbol{s}, a, \boldsymbol{s_z})$ is given by Equation 26 and is in this case independent of $\boldsymbol{s_z}$.

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

$$\boldsymbol{x}_i(\boldsymbol{s}, a) = (\boldsymbol{l}'_i, \boldsymbol{v}'_i, |\boldsymbol{v}_b|), \tag{27}$$

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 paper 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 24, where $\boldsymbol{\mu}_0$ is given by the expectation of Equation 26. Empirical evaluation of this monitor is given in Section 7.

7. EXPERIMENTS AND RESULTS

To empirically demonstrate our model correction framework, we implemented it on the CMDragons [2] team of soccer robots, which follow the specifications of the Small Size League of Robot Soccer. 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.

7.1 Evaluation Metrics

The first metric by which the model-correction framework was evaluated is Task Performance (TP). The ultimate goal of our monitor is to improve TP in environments with unmodeled behaviors, 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:

$$\text{TP}(\boldsymbol{\theta}) = \frac{n(\text{success}|\boldsymbol{\theta})}{n(\text{success}|\boldsymbol{\theta}) + n(\text{failure}|\boldsymbol{\theta})}. \tag{28}$$

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 $\boldsymbol{\theta}^0$:

$$\text{FRR}(\boldsymbol{\theta}^+, \boldsymbol{\theta}^0) = 1 - \frac{1 - \text{TP}(\boldsymbol{\theta}^+)}{1 - \text{TP}(\boldsymbol{\theta}^0)}. \tag{29}$$

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 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:

$$\text{MPA}(\boldsymbol{\theta}) = \frac{1}{|\boldsymbol{Z}|} \sum_{\boldsymbol{z}' \in \boldsymbol{Z}} P(\boldsymbol{z}|\boldsymbol{\theta}). \tag{30}$$

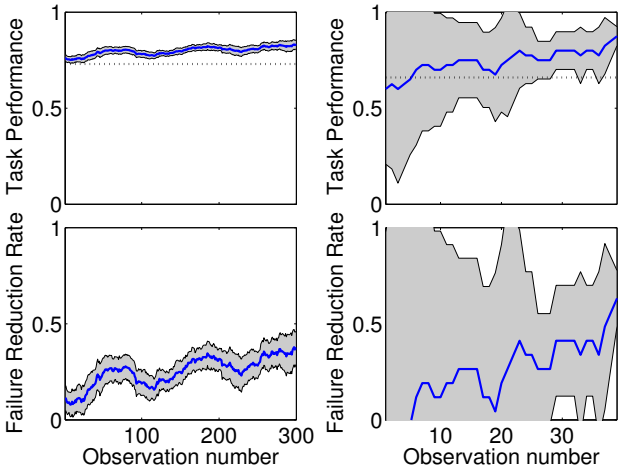
For all of our performance metrics, and throughout our experiments, we use as a baseline the original model $\boldsymbol{\theta}^0$.

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.

7.2 Experimental Results

First, we conducted extensive simulation tests to determine in a statistically significant way the evolution of TP($\boldsymbol{\theta}^+$)

and MPA($\boldsymbol{\theta}^+$) as the monitor acquires new observations. Figure 2a 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 FRR ≈ 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 similar trend, shown in Figure 2b. Baseline and adaptation performance were both higher in simulation than in the real robots, due to the complications added by the real world.



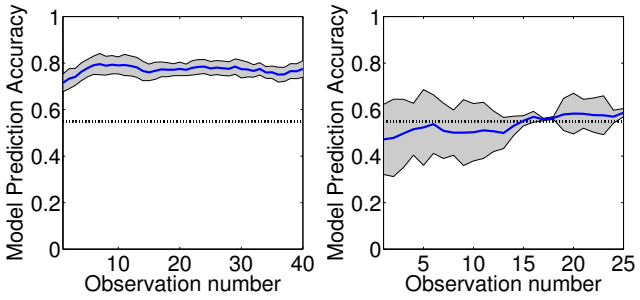
(a) Simulation (b) Real robots

Figure 2: 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.

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 regions, as this is where model improvement is expected to occur due to our monitor. For points that lie outside of the detected regions, the model (and thus MPA) remains unchanged by the monitor.

Figure 3a shows MPA($\boldsymbol{\theta}^+$) evaluation for simulation. With model correction, observations show a significantly better fit to the model than without correction. The ideal 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 MPA = 1, but to show significant improvement over the initial global model. Figure 3b shows less extensive real-robot test results, which show a similar trend as simulation.

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 subspaces



(a) Simulation (b) Real Robots

Figure 3: Moving average of prediction accuracy MPA($\boldsymbol{\theta}^+$), as a function of observations inside of anomalous subspaces. The shaded area shows the 95% confidence margin; the dotted black horizontal line indicates average baseline MPA($\boldsymbol{\theta}^0$)

may have attached semantic meaning 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 anomalous subspaces corresponded to flaws in the design of their simple prediction model of Equation 26. Some anomalous subspaces corresponded to inadequacies in the computation of navigation times t_k and t_t, while others corresponded to inadequacies in the assumption that the probability of success corresponds exclusively to a smoothly varying function of $t_t - t_k$. While this paper does not focus on the problem of assigning semantic meaning to discovered anomalous subspaces, it is an area of interest for future research.

8. CONCLUSION

This paper presented a framework for online robot adaptation in domains in which the robots have a generally accurate model of the effects of their actions, but in which there are particular sets of situations that do not conform to the model. The framework achieves adaptation from sparse observations by searching over parametric subspaces of the domain to find those in which observed behavior does not match expectations. After finding such subspaces, corrections are applied to the model to improve future predictions.

A key assumption that enables our framework to adapt from sparse observations is that anomalous behavior is restricted to specific subspaces of a state-action feature space, while the model is accurate in the rest of the domain. This assumption enables the monitor to run focused optimizations over potentially anomalous subspaces.

The monitor presented here was deployed in a realistic soccer-robot Keepaway domain, in which it significantly improved performance and model accuracy within timescales comparable to a single game of robot soccer. Furthermore, the anomalous subspaces detected by the monitor revealed to us flaws in the design of the interception model used by our soccer team; this will enable the model designers to correct such flaws for future competitive games.

9. ACKNOWLEDGMENTS

This material is based on work partially supported by the NSF Grant IIS-1012733, DARPA Grant FA87501220291, and MURI subcontract 138803 of Award N00014-09-1-1031. The presentation reflects only the views of the authors.

REFERENCES

[1] C. G. Atkeson and J. C. Santamaria. A comparison of direct and model-based reinforcement learning. In *In International Conference on Robotics and Automation*, 1997.

[2] J. Biswas, J. P. Mendoza, D. Zhu, B. Choi, S. Klee, and M. 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.

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

[4] R. J. Doyle, D. Atkinson, and R. Doshi. Generating perception requests and expectations to verify the execution of plans. In T. Kehler, editor, *AAAI*, pages 81–88. Morgan Kaufmann, 1986.

[5] C. Gaskett, D. Wettergreen, and A. Zelinsky. Q-learning in continuous state and action spaces. In *Australian Joint Conference on Artificial Intelligence*, pages 417–428. Springer-Verlag, 1999.

[6] G. D. Giacomo, R. Reiter, and M. Soutchanski. Execution monitoring of high-level robot programs. In *Principles of Knowledge Representation and Reasoning*, pages 453–465. Morgan Kaufmann, 1998.

[7] K. Z. Haigh and M. M. Veloso. Learning situation-dependent costs: Improving planning from probabilistic robot execution. In *Proceedings of the Second International Conference on Autonomous Agents*, AGENTS '98, pages 231–238, New York, NY, USA, 1998. ACM.

[8] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[9] E. Keogh and J. Lin. Hot sax: Efficiently finding the most unusual time series subsequence. In *ICDM*, pages 226–233, 2005.

[10] E. Keogh, S. Lonardi, and B. 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.

[11] M. Kulldorff. A spatial scan statistic. *Communications in Statistics-Theory and methods*, 1997.

[12] J. P. Mendoza, M. Veloso, and R. Simmons. Focused optimization for online detection of anomalous regions. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, June 2014.

[13] J. P. Mendoza, M. Veloso, and R. Simmons. Plan execution monitoring through detection of unmet expectations about action outcomes. In *Proceedings of the International Conference on Robotics and Automation (ICRA) (to appear)*, Seattle, USA, May 2015.

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

[15] O. Pettersson. Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems*, 53(2):73–88, Nov. 2005.

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

[17] P. Stone, G. Kuhlmann, M. E. Taylor, and Y. Liu. Keepaway soccer: From machine learning testbed to benchmark. In I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, editors, *RoboCup-2005: Robot Soccer World Cup IX*, volume 4020, pages 93–105. Springer Verlag, Berlin, 2006.

[18] P. Stone, R. S. Sutton, and G. Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

[19] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pages 1057–1063, 1999.