2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Online Parameter Estimation via Real-Time Replanning of Continuous Gaussian POMDPs

Dustin J. Webb Kyle L. Crandall Jur van den Berg

***Abstract*— An accurate dynamics model of a robot is an important ingredient of many algorithms used to solve robotics problems, including motion planning, control, localization, and mapping. Models derived from first principles often contain parameters (e.g. mass, moment of inertia, arm lengths, etc.) for which values are unknown. Those which cannot be easily measured must be estimated from the observed behavior of the robot. A good approach to address this problem is to plan control policies for the robot that elicit maximal amounts of information about the parameters of the system, while still achieving other objectives specified for the robot. In case of parameters subject to drift, this must be done continuously over the lifetime of the robot if costly re-calibrations are to be avoided. In this paper, we introduce a new method that formulates the parameter estimation problem as a continuous partially-observable Markov decision process (POMDP), which plans control policies that optimally trade-off the effort spent on learning parameters and effort spent on achieving regular robot objectives (exploration vs. exploitation), and allow for online, continual parameter estimation. While POMDPs have, until recently, been mostly of theoretical interest due to their inherent complexity, we build on recent advances that allow continuous, Gaussian POMDPs to be approximately-optimally solved in near-real-time rates. We show that the computed control policies lead to improved convergence of the belief of the parameters compared to system identification approaches based on applying random controls.**

I. Introduction

Many techniques in robotics require a dynamics model of a robot. Some examples include motion planning algorithms like RRT and its variants [1], [2], [3], planning algorithms like those found in [4] and [5], simultaneous localization and mapping (SLAM) [6], and many control algorithms. While a model can be derived from first principles this often leaves parameters that need to be estimated, such as component masses and moments of inertia, etc.

One major issue that arises when combining planning and learning is the exploration-exploitation problem. In other words, while maximizing an objective function, when should an agent exploit knowledge it has versus exploring strategies that may improve its knowledge and therefore lead to a reduction in cost over the long run? POMDPs naturally solve this problem as a result of the fact that they exhibit optimal information seeking behavior.

While using POMDPs has been of theoretical interest for quite some time their use in real-time applications has been considered impractical because, in the general form, POMDPs are computationally expensive [7]. However recent advances in calculating approximate solutions [4], [5], [8], [9], [10], [11] have brought this into the realm of possibility.

The authors are with the School of Computing and the Department of Mechanical Engineering at the University of Utah. E-mail: dustin.j.webb@utah.edu, kyle.crandall@utah.edu, berg@cs.utah.edu.

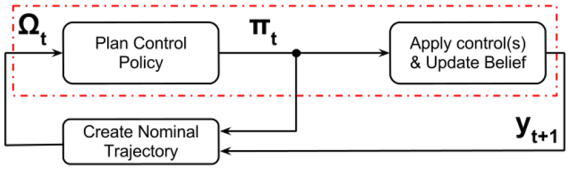


Fig. 1: Flow diagram of the general process of our approach. Given a nominal trajectory Ω_t, which includes an initial belief $\mathcal{Y}_t$, a control policy π_t is calculated and applied to the system (shown in dotted box). In the case of replanning only the first control is applied and the resulting belief $\mathcal{Y}_{t+1}$ is used in conjunction with the policy to generate a new nominal trajectory.

Here we introduce a method that efficiently estimates parameters by posing the problem of planning controls that elicit maximal amounts of information about the parameters as a partially observable Markov decision process (POMDP). This method uses the POMDP to calculate a control policy which yields, with each subsequent control applied, the greatest amount of information possible about the parameters while meeting any other objectives that need to be achieved. The control policy is then used to apply controls to the system and an extended Kalman filter is used to update the belief about the system. In our formulation the parameters are included in the state and therefore estimated simultaneously with it. This works so long as the real parameters are observable from the behavior of the system.

In this work we use the method developed in [4] and [5] which approximates beliefs as Gaussian distributions and the belief dynamics by an extended Kalman filter. This technique enables us to approximately solve continuous POMDPs in real-time which in turn allows for continually estimating the parameters through the life of the robot and prevents the need for possibly time consuming and costly re-calibrations.

We tested our method in simulation and with real hardware. Specifically we simulated a force controlled double integrator system as proof-of-concept. We performed experiments with an iRobot Create to show that our method does in fact work on real robots. Finally we simulated a 2 degree of freedom (DOF) planar arm to show that our method can find values for multiple parameters simultaneously.

The remainder of this paper is organized as follows. In the next section we discuss related work. In Section III we precisely define the problem we are addressing. We then outline our approach in Section IV. Results from our simulations and experiments are presented in Section V.

Section VI concludes with a discussion of the results and an outline of future work.

II. Related Work

Our work most closely aligns with system identification (SI) methods known as grey-box modeling methods because such methods estimate the parameters of a given model [12]. Many of these methods use statistical tools to determine the unknown parameters from observations made, and, if available, the controls applied to the system. These algorithms come in one of two forms; offline and online. Offline methods, such as expectation-maximization (EM) and Kalman smoothing, operate on data after it has been collected while online methods, such as Kalman filtering, perform the estimation while the system is functioning. One of the oldest and still highly popular grey-box modeling methods is to include model parameters in the state and use a Kalman filter [13], [14], [15], [16], and common variants like the extended Kalman filter [17] and the unscented Kalman filter [18], [19], to observe the behavior of the system as it executes some series of, often random, controls.

While our approach employs Kalman filtering in this same way, it goes further in that it plans control policies that yield the greatest information gain with respect to the parameters being estimated for each control executed.

Our approach is not the first to combine planning and learning. In fact there is a long and varied history to doing so. The earliest approaches date back to the work of Bellman [20] and Feldbaum [21]. Since their work this problem has been explored with many different methodologies including active learning [22], reinforcement learning [23], [24], adaptive control [25], neurodynamic control [26], evolutionary approaches [27], and model predictive control [28]. Of course these approaches are not necessarily focused on finding the unknown parameters to a given model.

Our approach is not even the first to plan control policies via POMDPs. Planning in a belief space while assuming maximum likelihood observations was introduced in [29]. Similarly [30] explores planning in a belief space but focuses on planning in large spaces over long periods of time. More recently the idea of calculating approximate solutions was explored in [4], [5], and even more recently it was shown in [31] that optimal online learning can be achieved by using a POMDP to plan policies offline that direct an agent to optimally balance exploration and exploitation during execution.

Our method differs from that of [31] in two important ways. First, the method of [31] requires discretization of the state-space which limits its scalability. This follows from the fact that discretizing a continuous state-space results in an exponential growth in the number states therefore inflicting such methods with the *curse of dimensionality*. Second, our method is targeted at real-time applications through rapid replanning which permits it to adapt to unexpected events.

III. Preliminaries and Definitions

In the following subsections we review the basics of POMDPs and extended Kalman filters (EKFs). From there we briefly summarize the specific method that we use for solving POMDPs. Then we define parameter and augmented-state spaces. Finally we formally state the problem we are addressing in this paper.

A. Partially Observable Markov Decision Processes

POMDPs are a principled method for planning under uncertainty [6]. They build on Markov Decision Processes (MDPs) by taking into account that the state of the world is not directly or completely observable.

Formally a POMDP consists of a set $\mathbb{X}$ of all possible states a system can take, a set $\mathbb{U}$ representing all possible controls that can be applied to the system, a set $\mathbb{Z}$ describing all possible observations that can be made of the system. A POMDP also requires stochastic transition and observation models. The stochastic nature of these models leads to a set of beliefs $\mathbb{B}$ where $\mathcal{X} \in \mathbb{B}$ is a probability distribution over a state given all past inputs and observations, and a function $\beta : \mathbb{B} \times \mathbb{U} \times \mathbb{Z} \to \mathbb{B}$ known as a Bayesian filter, or the belief dynamics, that describes how beliefs evolve as a function of the current belief, the control applied to that belief, and the observation acquired after applying the control:

$$\mathcal{X}_t = \beta(\mathcal{X}_{t-1}, \mathbf{u}_{t-1}, \mathbf{z}_t). \tag{1}$$

A *finite horizon POMDP* is used when an agent is limited to a finite number T of sequential actions. The objective is to derive policies $\pi_t : \mathbb{B} \to \mathbb{U}$ for all $0 \leq t < T$, referred to collectively as the control policy π, such that applying controls $\mathbf{u}_t = \pi_t(\mathcal{X}_t)$ minimizes the expected value of an objective function:

$$\mathrm{E}_{\mathbf{z}_1,\ldots,\mathbf{z}_T}[c_T(\mathcal{X}_T) + \sum_{t=0}^{T-1} c_t(\mathcal{X}_t, \mathbf{u}_t)], \tag{2}$$

where $c_T : \mathbb{B} \to \mathbb{R}$ and $c_t : \mathbb{B} \times \mathbb{U} \to \mathbb{R}$ are immediate cost functions and are considered inputs to the POMDP problem.

Value iteration is a general approach to solving finite horizon POMDPs. It works by iterating backwards from step T and calculating the *value function* which maps beliefs to the expected cost-to-go at any given step t, i.e. $v_t : \mathbb{B} \to \mathbb{R}$:

$$v_T(\mathcal{X}) = c_T(\mathcal{X}), \tag{3}$$

$$v_t(\mathcal{X}) = \min_{\mathbf{u} \in \mathbb{U}} (c_t(\mathcal{X}, \mathbf{u}) + \mathrm{E}_{\mathbf{z}}[v_{t+1}(\beta(\mathcal{X}, \mathbf{u}, \mathbf{z}))]). \tag{4}$$

The optimal control $\mathbf{u}_t$ given a belief $\mathcal{X}_t$ is then the one that minimizes the minimand of Eq. (4) at step t:

$$\pi_t[\mathcal{X}] = \operatorname*{argmin}_{\mathbf{u} \in \mathbb{U}} (c_t(\mathcal{X}, \mathbf{u}) + \mathrm{E}_{\mathbf{z}}[v_{t+1}(\beta(\mathcal{X}, \mathbf{u}, \mathbf{z}))]). \tag{5}$$

B. Extended Kalman Filter

The EKF is a Bayesian filter for Gaussian beliefs $\mathcal{X} \sim \mathcal{N}(\hat{\mathbf{x}}, \Sigma)$. It adapts the Kalman filter to systems with non-linear transition and observation models of the form:

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{m}_t, \quad \mathbf{m}_t \sim \mathcal{N}(\mathbf{0}, M(\mathbf{x}_t, \mathbf{u}_t)), \tag{6}$$

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \mathbf{n}_t, \quad \mathbf{n}_t \sim \mathcal{N}(\mathbf{0}, N(\mathbf{x}_t)), \tag{7}$$

where $\mathbf{m}_t$ is state and control dependent Gaussian distributed noise related to the transition model and, similarly, $\mathbf{n}_t$ is

state dependent Gaussian distributed noise related to the observation model.

Given a belief $\mathcal{X}_t \sim \mathcal{N}(\hat{\mathbf{x}}_t, \Sigma_t)$, a control $\mathbf{u}_t$, and an observation $\mathbf{z}_{t+1}$, Eq. (1) is approximated to calculate $\mathcal{X}_{t+1} \sim \mathcal{N}(\hat{\mathbf{x}}_{t+1}, \Sigma_{t+1})$ by the EKF update equations

$$\hat{\mathbf{x}}_{t+1} = \mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t) + K_t(\mathbf{z}_{t+1} - \mathbf{h}(\mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t))), \quad (8)$$

$$\Sigma_{t+1} = (I - K_t H_t)\Gamma_t, \quad (9)$$

where

$$\Gamma_t = A_t \Sigma_t A_t^T + M(\hat{\mathbf{x}}_t, \mathbf{u}_t), \quad (10)$$

$$K_t = \Gamma_t H_t^T (H_t \Gamma_t H_t^T + N(\mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t)))^{-1}, \quad (11)$$

$$A_t = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\hat{\mathbf{x}}_t, \mathbf{u}_t), \quad H_t = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}(\mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t)), \quad (12)$$

and A_t, H_t represent linearizations of the transition and observation models about the maximum likelihood state.

C. Approximate Locally Optimal POMDP

It is well known that exact solutions to POMDPs are computationally complex [7]. However, recent work [4], [5] has shown that in the case of continuous state, control, and observation spaces (i.e $\mathbb{X} = \mathbb{R}^n$, $\mathbb{U} = \mathbb{R}^m$, and $\mathbb{Z} = \mathbb{R}^p$), and sufficiently smooth transition and observation models of the form in Eqs. (6) and (7), and for which the Gaussian distribution forms a reasonable approximation of the belief, i.e. $\mathcal{X} = \mathcal{N}(\hat{\mathbf{x}}, \Sigma)$, locally optimal solutions can be computed efficiently. Further, it takes into account the fact that the measurements being unknown in advance means the belief dynamics are stochastic. The net result is that the belief dynamics given a belief $\mathcal{X}_t$ become

$$\begin{bmatrix} \hat{\mathbf{x}}_{t+1} \\ \Sigma_{t+1} \end{bmatrix} = \Phi(\hat{\mathbf{x}}_t, \Sigma_t, \mathbf{u}_t) + \mathbf{w}_t, \quad (13)$$

with $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, W(\hat{\mathbf{x}}_t, \Sigma_t, \mathbf{u}_t))$,

$$\Phi(\hat{\mathbf{x}}_t, \Sigma_t, \mathbf{u}_t) = \begin{bmatrix} \mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t) \\ (I - K_t H_t)\Gamma_t \end{bmatrix}, \quad (14)$$

$$W(\hat{\mathbf{x}}_t, \Sigma_t, \mathbf{u}_t) = \begin{bmatrix} K_t H_t \Gamma_t & 0 \\ 0 & 0 \end{bmatrix}, \quad (15)$$

and Γ_t, K_t, and H_t are given by Eqs. (10), (11), and (12) respectively.

Besides the stochastic transition and observation models this method requires local cost functions $c_t : \mathbb{B} \times \mathbb{U} \to \mathbb{R}$ and $c_T : \mathbb{B} \to \mathbb{R}$ which can be reasonably approximated by quadratization with respect to the belief. These cost functions form the basis of the objective function (see Eq. (2)) to be optimized.

The method requires an initial belief $\mathcal{X}_0 = \mathcal{N}(\hat{\mathbf{x}}_0, \Sigma_0)$ and an arbitrary nominal trajectory to be given of the form consistent with the deterministic part of the belief dynamics:

$$\Omega = (\bar{\mathbf{x}}_0, \bar{\Sigma}_0, \mathbf{u}_0, ..., \bar{\mathbf{x}}_{T-1}, \bar{\Sigma}_{T-1}, \mathbf{u}_{T-1}, \bar{\mathbf{x}}_T, \bar{\Sigma}_T). \quad (16)$$

Additionally it uses an *iterative* approach based on linear quadratic approximations of the belief dynamics and cost functions about the nominal trajectory that converges to a locally optimal control policy of the linear form:

$$\mathbf{u}_t = \pi_t(\mathcal{X}_t) = L_t \begin{bmatrix} \hat{\mathbf{x}}_t \\ \text{vec}[\Sigma_t] \end{bmatrix} + \mathbf{l}_t, \quad (17)$$

for $0 \leq t < T$. Details for implementing the algorithm can be found in [4] and [5].

Using an approximate representation of the value function that is quadratic in the belief means that a locally optimal control policy can be calculated with a running time polynomial in the dimension n of the state-space ($O(n^4)$) and opens the door to applying POMDPs to real-time applications.

D. Parameter and Augmented Spaces

Given transition and observation models with k parameters whose values are unknown or of which one only has a rough estimate, we define a *parameter-space* $\mathbb{K} = \mathbb{R}^k$ as the space of all possible parameter combinations.

We assume the unknown parameters are subject to drift as modeled by stochastic white noise. As such the transition model is:

$$\mathbf{k}_{t+1} = \mathbf{k}_t + \boldsymbol{\theta}_t, \quad \boldsymbol{\theta}_t \sim \mathcal{N}(\mathbf{0}, \Theta), \quad (18)$$

where $\mathbf{k}_t \in \mathbb{K}$, and $\boldsymbol{\theta}_t$ is a noise term drawn from a zero mean Gaussian distribution with covariance Θ.

We compose an *augmented-space* vector $\mathbf{y} \in \mathbb{Y} = \mathbb{X} \times \mathbb{K}$ by appending a parameter-space vector to a state-space vector as follows:

$$\mathbf{y} = \begin{bmatrix} \mathbf{x} \\ \mathbf{k} \end{bmatrix}, \quad \mathbf{x} \in \mathbb{X}, \quad \mathbf{k} \in \mathbb{K}. \quad (19)$$

The augmented-space transition and observation models are then:

$$\mathbf{y}_{t+1} = \begin{bmatrix} \mathbf{f}(\mathbf{y}_t, \mathbf{u}_t) \\ \mathbf{k}_t \end{bmatrix} + \begin{bmatrix} \mathbf{m}_t \\ \boldsymbol{\theta}_t \end{bmatrix}, \quad (20)$$

$$\mathbf{z}_t = \mathbf{h}(\mathbf{y}_t) + \mathbf{n}_t, \quad (21)$$

with $\mathbf{m}_t$, $\mathbf{n}_t$, and $\boldsymbol{\theta}_t$ being Gaussian distributed noise as defined for Eqs. (6), (7), and (18) and

$$\mathbf{f}(\mathbf{y}_t, \mathbf{u}_t) = \mathbf{f}(\mathbf{x}_t, \mathbf{k}_t, \mathbf{u}_t), \quad \mathbf{h}(\mathbf{y}_t) = \mathbf{h}(\mathbf{x}_t, \mathbf{k}_t). \quad (22)$$

In other words the transition model is a function of the state $\mathbf{x}_t$ and control $\mathbf{u}_t$ as in Eq. (6) but also the parameters to be estimated $\mathbf{k}_t$. Similarly the observation model becomes a function of the parameters $\mathbf{k}_t$ in addition to the state $\mathbf{x}_t$. Note that we will refer to beliefs associated with augmented-space vectors as $\mathcal{Y}$.

E. Problem Definition

Our method addresses the question of how to compute control polices that yield maximal amounts of information about parameters to be estimated. That is control policies that when employed 1) result in behavior from which the true parameter values of the system can be inferred, 2) can be inferred quickly, and 3) achieve any other objectives encoded in the objective function.

In this paper we consider controllable systems with continuous stochastic and possibly non-linear transition and

observation models of the form denoted by Eqs. (6) and (7) with k unknown but observable parameters. Given an initial belief $\mathcal{Y}_t$, these models are used to form augmented-space transition and observation models of the form in Eqs. (20) and (21).

We assume the state, control, observation, and parameter spaces to be continuous (i.e. $\mathbb{X} = \mathbb{R}^n$, $\mathbb{U} = \mathbb{R}^m$, $\mathbb{Z} = \mathbb{R}^p$, and $\mathbb{K} = \mathbb{R}^k$) and that the belief dynamics are reasonably approximated by an EKF.

The objective is to minimize the variance in the belief of the parameters, i.e. learn the values for these parameters, while at the same time achieving any other objective that may be specified for the system, e.g. reaching a particular goal state. We do this by seeking control polices $\mathbf{u}_t = \pi_t(\mathcal{Y})$ for $0 \leq t < T$ that minimize Eq. (2) for cost functions that penalize the state of the robot and the variance of the parameters. In other words, functions of the following form, though any function quadratic in the belief, will work.

$$c_T(\hat{\mathbf{y}}, \Sigma) = (\hat{\mathbf{x}} - \mathbf{x}^*)^T Q_T (\hat{\mathbf{x}} - \mathbf{x}^*) + \text{tr}(S_T \Sigma_{\mathbf{x}}) + \text{tr}(T_T \Sigma_{\mathbf{k}}), \tag{23}$$

$$c_t(\hat{\mathbf{y}}, \Sigma, \mathbf{u}) = (\hat{\mathbf{x}} - \mathbf{x}^*)^T Q_t (\hat{\mathbf{x}} - \mathbf{x}^*) + \mathbf{u}^T R_t \mathbf{u} + \text{tr}(S_t \Sigma_{\mathbf{x}}) + \text{tr}(T_t \Sigma_{\mathbf{k}}), \tag{24}$$

for $Q_t \geq 0$ which penalizes the deviation of the state-space elements $\hat{\mathbf{x}}$ of the augmented-space vector $\hat{\mathbf{y}} = [\hat{\mathbf{x}}^T, \hat{\mathbf{k}}^T]^T$ from the desired goal state $\mathbf{x}^*$, $S_t, T_t \geq 0$ which penalize the state-space elements $\Sigma_{\mathbf{x}}$ and parameter-space elements $\Sigma_{\mathbf{k}}$ respectively of the augmented-space covariance matrix $\Sigma = \left[\begin{smallmatrix} \Sigma_{\mathbf{x}} & \Sigma_{\mathbf{xk}} \\ \Sigma_{\mathbf{kx}} & \Sigma_{\mathbf{k}} \end{smallmatrix}\right]$, and $R_t > 0$ which penalizes the magnitude of the controls.

The penalty matrices Q, R, S, and T define the relative importance between precisely reaching the goal, constraining the control magnitudes, and seeking information about the parameters. R must be positive-definite while the other matrices must be either zero or positive semidefinite. If the norms of Q and T are greater than 0 then the control policies generated will seek information about the parameters to be estimated but only in so far as is necessary to meet the other objectives.

IV. Approach

In this section we outline our approach. We begin by describing the basics of our approach. From there we discuss real-time use of our method through continual replanning. Finally we highlight details that need to be considered when implementing our method.

A. Basic Approach

At a high level the most basic implementation of our method is to plan a single control policy and use it to control the robot. Each control applied informs our belief as to the robots state and the parameters to be estimated leaving us with an improved estimate of the parameters. This is the portion of the diagram outlined by the dashed line in Fig. 1.

Given an arbitrary nominal trajectory, a continuous Gaussian POMDP as formulated in Section III-C is used to plan a control policy. The control policy is then used to determine the controls to be applied to the system where the initial belief estimate is used to calculate the first control. An EKF is simultaneously used to update the belief which is used to calculate the next control in turn. Being an augmented-space state, estimates of the parameters are included. If the parameters are observable from the behavior of the system then parameter estimates will automatically be updated as will the variance of the parameters which gives an indication of the uncertainty about the parameters.

EstimateParamsWithReplanning($\mathcal{Y}_0, \Omega_0$)

for $t \in [0, T)$ **do**
  $\{L_i, \mathbf{l}_i\}_{i=t}^{T-1} \leftarrow$ SolvePOMDP(Ω_t)
  $\bar{\mathbf{u}}_t \leftarrow L_t \begin{bmatrix} \hat{\mathbf{x}}_t \\ \text{vec}[\Sigma_t] \end{bmatrix} + \mathbf{l}_t$
  ApplyControl($\bar{\mathbf{u}}_t$)
  $\mathbf{z}_{t+1} \leftarrow$ ReceiveMeasurement()
  $\mathcal{Y}_{t+1} \leftarrow$ UpdateBelief($\mathcal{Y}_t, \bar{\mathbf{u}}_t, \mathbf{z}_{t+1}$)
  $\Omega_{t+1} \leftarrow$ UpdateNomTraj($t+1, \mathcal{Y}_{t+1}, \{L_i, \mathbf{l}_i\}_{i=t+1}^{T-1}$)
return $\{\mathcal{Y}_i\}_{i=0}^{T}$

Fig. 2: Pseudocode for an implementation of our method that replans the control policy after each control applied. this has the benefit of improving the control policy as the robot operates.

B. Continual Replanning Approach

In the basic approach the control policy is calculated based on the initial belief alone. As such the control policies are locally optimal with respect to some nominal trajectory that is necessarily based on highly uncertain parameter estimates. These uncertainties lead to suboptimal control policies and as the length of the trajectory increases the errors that arise compound.

This issue can be addressed by taking into account the data we acquire at each time step to further inform the control policy through rapid replanning. In other words, at each step and before a control is applied to the system, we calculate a new control policy, for that step and beyond. We use our current control policy and our current belief to generate a new nominal trajectory from which the new control policy will be derived, a process we refer to as a *warm start* because it begins with a control policy that is likely to be similar to the new control policy.

Since the POMDP approach is iterative it is expected to converge quickly given a warm start, making it truly applicable to real-time application. However, convergence can still take too long under certain conditions. This is especially the case when the stopping criteria is overly strict, when the control frequency is high, or when the parameters estimates suddenly change significantly, a scenario which occurs if the initial parameter estimates are far from the true parameter values. To address this issue we use an anytime version of the POMDP algorithm wherein we stop the iterative process as soon as time runs out, meaning a new control must be applied, and use the latest control policy to determine the next control.

Once a new control policy has been calculated the algorithm proceeds as before; a control is calculated and applied, a measurement received, the belief updated, and a new control policy computed, etc.. This proceeds for T steps for episodic tasks which means the number of control policies planned at each step becomes smaller with each step. For use on long-term autonomous robots T control policies are calculated with each iteration, as is common with model predictive controllers.

C. Implementation details

A number of considerations need to be made when implementing our method, from selecting initial controls to extracting the parameter estimates from the belief data. These considerations are outlined here.

1) Initial Controls: The initial nominal trajectory is based on an initial belief and a series of controls. A common approach to generating such controls is to use the initial estimate of the model with a motion planning algorithm like RRT. Of course, this requires a specific goal state. If the objective is strictly to estimate the parameters of the model then the initial controls can be generated at random.

2) Constraints and Obstacles: All real systems have constraints. Control saturation is one such example. Controls can be penalized through the cost functions in Eqs. (23) and (24) however this does not place a hard bound on the controls. As such the control policy calculated by our method can yield controls that are beyond the capabilities of the system. To handle this, one can saturate the controls. While doing so is inconsistent with the model, experiments show it works well.

Constraints also appear as bounds on the state, such as maximum velocity, and as physical obstacles. The POMDP approach we use allows for encoding such constraints into the objective function. Details can be found in [4] and [5].

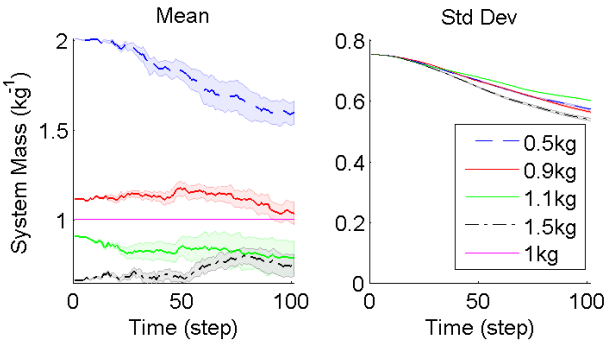
3) Extracting Parameter Estimates: Each belief in the sequence generated by our method include an estimate of the parameters. The naive approach to extracting these values would be to take them directly from the last belief $\mathcal{Y}_T$ in the case of episodic usage and the most recent belief $\mathcal{Y}_t$ in the case of long-term applications. However given the presence of noise in the dynamics a better approach may be to calculate a moving average over some window of the last few beliefs, i.e.

$$\tilde{\mathbf{k}} = \frac{1}{n} \sum_{i=t-j}^{t} \hat{\mathbf{k}}_i, \tag{25}$$

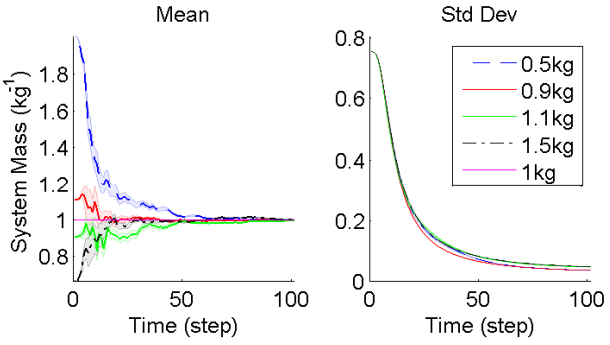
where $\hat{\mathbf{k}}_t$ comes from the augmented-space vector $\hat{\mathbf{y}}_t$ as defined in Eq. (19), t is the most recent step, and $n = t - j$ defines a window size.

V. Simulation and Experiment Results

We tested our approach with three systems. The first was a simulation of the classic double integrator model as a proof of concept to the idea. The second was a differential drive model which allowed us to show the idea works with real robots. The third and final model was a simulated 2 DOF planar arm to show that our method can learn multiple parameters. We compare our method to that of applying random controls to a robot and estimating the parameters via an EKF, a technique frequently used with SI methods. The results of our simulations and experiments are discussed in the following subsections.



(a) Random Controls



(b) POMDP Controls

Fig. 3: Comparison of random controls with those planned via POMDP for double integrator model with actual mass $\mu = 1$kg. The inverse of the mass μ^{-1} was learned to improve numerical stability and four different initial values were tested; 0.5kg, 0.9kg, 1.1kg, and 1.5kg.

A. Double Integrator Model

The first system we used to test our approach was a force controlled 2D double integrator. By using force control we introduced a learnable parameter, namely the mass of the robot. The parameter-space model for this system is defined:

$$\mathbf{y}_{t+1} = A\mathbf{y}_t + B\mathbf{u}_t + m_t, \quad \mathbf{z}_t = H\mathbf{y}_t + n_t, \tag{26}$$

with $\mathbf{y} = [\mathbf{p}^T, \mathbf{v}^T, \mu]^T$,

$$A = \begin{bmatrix} I & I\Delta t & \mathbf{0} \\ \mathbf{0} & I & \mathbf{0} \\ \mathbf{0}^T & \mathbf{0}^T & 1 \end{bmatrix}, \quad B = \begin{bmatrix} I\Delta t^2/2\mu \\ I\Delta t/\mu \\ 0 \end{bmatrix}, \quad H = \begin{bmatrix} I, \mathbf{0} \end{bmatrix}, \tag{27}$$

where $\mathbf{y}$ is an augmented-space vector, μ is the unknown mass parameter, I is the 2×2 identity matrix, Δt is the time step between controls, and at any given time t, p_x, p_y, v_x, and v_y are the x and y coordinates and respective instantaneous velocities of the robot.

Though the double integrator is normally a linear system it becomes non-linear when we convert it to force control because μ appears in the control matrix B.

Given that μ appears in both the position and velocity components of the transition model most actions are informative. However higher magnitude controls produce larger movements over a given time delta which in effect reduces the impact of the motion noise.

This fact permits us to show that the POMDP seeks out more informative controls. First we generate nominal controls from a zero-mean Gaussian distribution with unit variance. For a 1 kg mass such controls will produce movement and thus information regarding the mass of the robot. Figure 3a shows the results of using such controls. Figure 3b shows the results of using these controls to generate a nominal trajectory for planning a control policy and using it in turn to control the robot.

Each method was executed 30 times. The left graph for each method shows the average of the mean belief as a function of time. The right graphs show the average standard deviation of the belief at each step. Each graph also includes the 98% confidence interval which describes the variation in the 30 experiments.

From Figure 3a we can see that the random controls generally result in updates of the mean of the belief of the parameter towards the correct value. They also consistently reduce the average standard deviation on the belief. On the other hand, as we see in Figure 3b, the controls from the POMDP result in much swifter convergence, and move the belief towards the correct value regardless of the initial estimate. We also see that the POMDP controls serve to reduce the standard deviation considerably more in comparison and does so at a faster rate which indicates that our method becomes confident of its estimate in fewer control steps.

B. Differential Drive

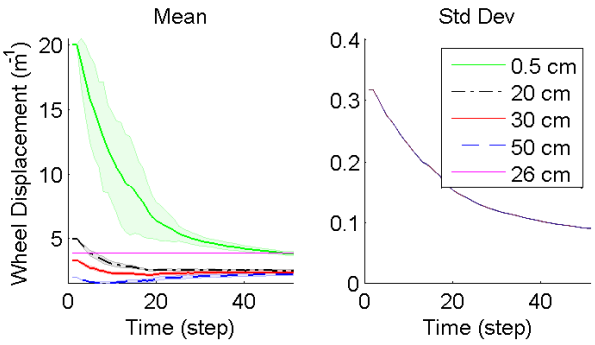
We used an iRobot Create to test our method in the real world. This is a two-wheel differential drive robot. As such our state contains the 2D location of the robot and its heading, i.e. $\mathbf{x} = [x, y, \theta]^T$, which was measured via a motion capture system. The parameter to be learned is the distance between the wheels, i.e. $k = L$. The real value for this parameter was estimated to be 26cm. The control for the Create is a two dimensional vector comprised of a linear velocity for each of the wheels $\mathbf{u} = [u_r, u_l]^T$.

The continuous model for this system is:

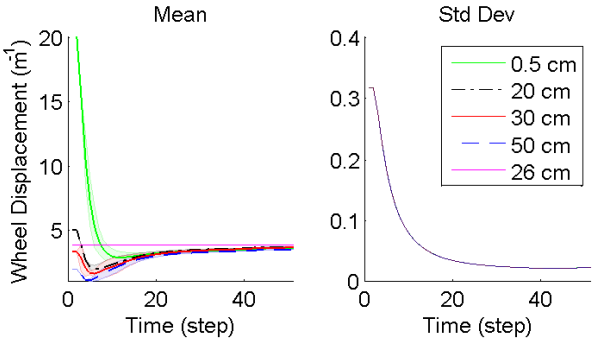
$$\dot{x} = \frac{u_r + u_l}{2}\cos\theta,\ \dot{y} = \frac{u_r + u_l}{2}\sin\theta,\ \dot{\theta} = \frac{u_r - u_l}{L}. \quad (28)$$

Since this is a continuous model we discretize it using Runge-Kutta (RK4) to form the discrete model $\mathbf{f}(\mathbf{y}_t, \mathbf{u}_t)$.

It is obvious from the model that only movements involving some amount of rotation are informative with respect to L. Inspection of the control space reveals that $u_l = u_r$ produces pure translations while $u_l = -u_r$ produces pure rotations. Furthermore all other controls result in a combination of translation and rotation except the trivial case where $u_l = u_r = 0$ which produces no movement at all.



(a) Random Controls



(b) POMDP Controls

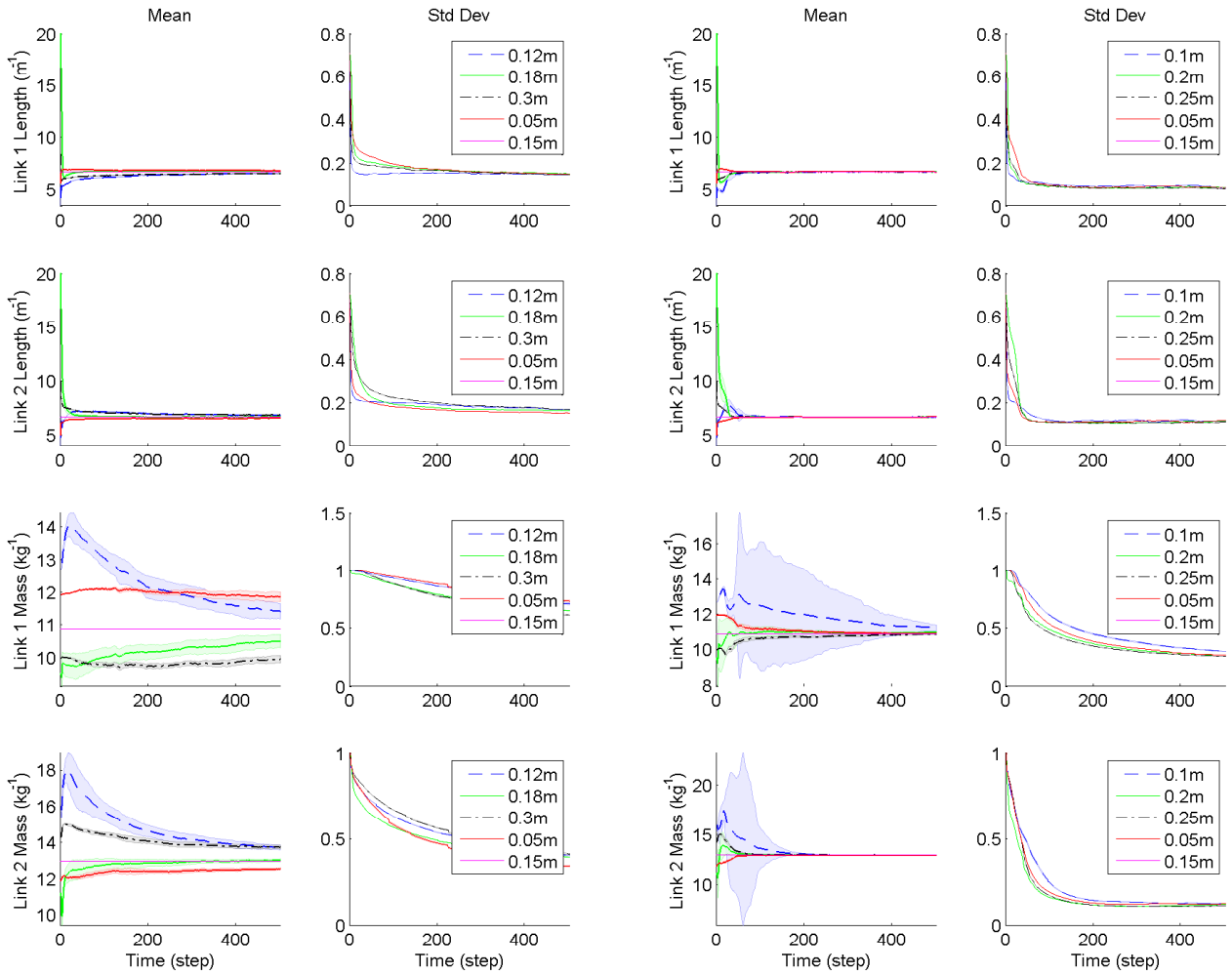
Fig. 4: Comparison of random controls with those planned via POMDP for differential drive system with actual wheel displacement $L = 26$cm. The inverse of the length L^{-1} was learned to improve numerical stability and four different initial values were tested; 0.5cm, 20cm, 30cm, and 50cm.

From these observations it is clear that the vast majority of controls selected at random will be informative. This comes from the fact that predictions of such movements while using the wrong parameter will differ from what actually happens. Similarly, larger controls will produce larger differences and will be less burdened by the noise in the translation and sensor models. As such we can reason that the optimal movements are the pure clockwise and counter-clockwise rotations executed at full velocity.

Figure 4 shows how the belief develops using random controls versus those generated by our approach. As with the double integrator system we see that learning does occur when using the random controls but that using the POMDP controls results in much quicker convergence in the mean of the belief and a greater decrease in the average standard deviation of the belief.

C. 2 DOF Planar Arm

The final system we used to test our approach was a simulation of the Quanser SRV02 configured as a 2 DOF planar arm. This system is highly non-linear and contains many parameters making it a great candidate for showing that our method can learn multiple interrelated parameters simultaneously. The state is comprised of the angular positions and velocities of each joint, i.e. $\mathbf{x} = [\boldsymbol{\theta}^T, \dot{\boldsymbol{\theta}}^T]^T$ and the controls are the currents applied to each motor $\mathbf{u} = [i_1, i_2]^T$. We seek to learn the lengths and masses for each link in the



(a) Random Controls

(b) POMDP Controls

Fig. 5: Comparison of random controls with those planned via POMDP. To improve numerical stability, the inverse of the length a_i^{-1} and mass m_i^{-1} was learned for each of the two links $i \in \{1, 2\}$ in the 2 DOF planar arm. We ran the approach with four different initial values with initial standard deviation of 1m for the lengths and 1kg for the masses: (0.12 m, 0.1 m, 0.1 kg, 0.07 kg), (0.18 m, 0.2 m, 0.084 kg, 0.084 kg), (0.3 m, 0.25 m, 0.079 kg, 0.065 kg), (0.05 m, 0.05 0 m, 0.105 kg, 0.089 kg). The actual values were (0.15 m, 0.15 m, 0.092 kg, 0.077 kg).

arm, i.e. $\mathbf{k} = [a_1, a_2, m_1, m_2]^T$. The true length of each link is 15cm while the true weights are 92g for the upper arm link and 77g for the lower arm link including all the hardware connecting them. Finally, the observation model consists of the position of the end effector and the angular position of each joint.

For this robot we used the continuous current control model:

$$\begin{bmatrix} N_1 k_{t1} i_1 \\ N_2 k_{t2} i_2 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \ddot{\phi}_1 \\ \ddot{\phi}_2 \end{bmatrix} + \begin{bmatrix} 0 & -h \\ h & 0 \end{bmatrix} \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{bmatrix} + \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} + \begin{bmatrix} F_1 \\ F_2 \end{bmatrix}, \quad (29)$$

where

$$\begin{aligned} F_1 &= b_1 \dot{\phi}_1 + c_1 \text{sgn}(\dot{\phi}_1), \\ F_2 &= b_2 \dot{\phi}_2 + c_2 \text{sgn}(\dot{\phi}_2), \\ H_{11} &= N_1^2 J_{m1} + I_1 + m_2 a_1^2, \\ H_{22} &= N_2^2 J_{m2} + I_2, \\ G_1 &= (r_{01} m_2 + a_1 m_2) g \cos(\phi_2)), \\ G_2 &= r_{12} m_2 g \cos(\phi_2), \\ H_{12} &= H_{21} = a_1 r_{12} m_2 \cos(\phi_2 - \phi_1), \end{aligned}$$

and for $n \in \{1, 2\}$, a_n represents link lengths, m_n represents link masses, $r_{n-1,n}$ represents link center of mass, I_n represents link moment of inertia, J_{mn} represents motor inertia, b_n represents viscous damping, c_n represents coulomb friction, and N_n represents gear ratio. Since this is a continuous

model we discretize it using Runge-Kutta (RK4) to form $\mathbf{f}(\mathbf{y}_t, \mathbf{u}_t)$ as with the model of the iRobot Create.

Figure 5 compares our approach with that of applying random controls to the arm. Each figure has two graphs for each of the link masses and lengths. The left graph shows the average of the mean of the belief and the right shows the average of the standard deviation of the belief over 30 trials. Each graph also shows the 98% confidence intervals over the trials.

As with the other systems we see that random controls do result in learning and reduction in the standard deviation of the belief. However our method converges in all cases, and considerably more quickly in most of them, while using random controls generally does not converge to correct values. Of particular interest here is that both methods perform well when learning the link lengths. However when it comes to the masses its clear again that our method learns more quickly implying the masses are harder to learn.

VI. Discussion, Conclusion, and Future Work

In this paper we introduced a method for planning optimal learning control policies in real-time using an approximate POMDP solver. These policies identify controls that are approximately optimal for learning one or more unknown parameters associated with the model of a robot while respecting any other objectives encoded in the cost functions. Further, it does so in a manner that approximately optimally solves the exploration-exploitation problem and operates fast enough for use in real-time applications.

From the simulation and experimental results it is clear that our method effectively learns the unknown parameters of a robot model as the parameter estimates converge quickly to the real parameter values when using our method, especially as compared to using random controls. While this serves as a proof-of-concept there is still plenty of work ahead.

Going forward we will be exploring the limits of our method. We will do so by comparing it to other methods, by assessing the effects of the assumptions we make and the approximations we use, by simulating and experimenting with more complicated models that include more parameters and parameters that change over time, and by delving more into the rapid replanning to determine when it is useful versus when it is necessary. In conclusion, the work we have started in this paper is a major step towards a paradigm of life-long calibration.

References

[1] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[2] R. Tedrake, "Lqr-trees: Feedback motion planning on sparse randomized trees," 2009.

[3] D. J. Webb and J. van den Berg, "Kinodynamic rrt*: Optimal motion planning for systems with linear differential constraints," in *Proc. IEEE Conf. on Robotics and Automation*, 2013.

[4] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1263–1278, 2012.

[5] J. van den Berg, S. Patil, and R. Alterovitz, "Efficient approximate value iteration for continuous gaussian pomdps." in *AAAI*, 2012.

[6] S. Thrun, W. Burgard, D. Fox *et al.*, *Probabilistic robotics*. MIT press Cambridge, 2005, vol. 1.

[7] C. Papadimitriou and J. N. Tsitsiklis, "The complexity of markov decision processes," *Math. Oper. Res.*, vol. 12, no. 3, pp. 441–450, Aug. 1987.

[8] H. Kurniawati, D. Hsu, and W. S. Lee, "Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces." in *Robotics: Science and Systems*, 2008, pp. 65–72.

[9] T. Smith and R. G. Simmons, "Point-based POMDP algorithms: Improved analysis and implementation," in *Proc. Int. Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2005.

[10] Y. Du, D. Hsu, H. Kurniawati, W. Lee, S. Ong, and S. Png, "A pomdp approach to robot motion planning under uncertainty," in *Int. Conf. on Automated Planning Scheduling, Workshop on Solving Real-World POMDP Problems*, 2010.

[11] L. L. Ko, D. Hsu, W. S. Lee, and S. C. Ong, "Structured parameter elicitation." in *AAAI*, 2010.

[12] K. Keesman, *System Identification: An Introduction*, ser. Advanced textbooks in control and signal processing. Springer, 2011.

[13] A. H. Jazwinski, *Stochastic processes and filtering theory*. Courier Dover Publications, 2007.

[14] V. Aidala, "Parameter estimation via the kalman filter," *Automatic Control, IEEE Transactions on*, vol. 22, no. 3, pp. 471–472, 1977.

[15] J. N. Nielsen, H. Madsen, and P. C. Young, "Parameter estimation in stochastic differential equations: an overview," *Annual Reviews in Control*, vol. 24, pp. 83–94, 2000.

[16] N. R. Kristensen, H. Madsen, and S. B. Jørgensen, "Parameter estimation in stochastic grey-box models," *Automatica*, vol. 40, no. 2, pp. 225–237, 2004.

[17] L. Ljung, "Asymptotic behavior of the extended kalman filter as a parameter estimator for linear systems," *Automatic Control, IEEE Transactions on*, vol. 24, no. 1, pp. 36–50, 1979.

[18] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*. IEEE, 2000, pp. 153–158.

[19] R. Van Der Merwe and E. A. Wan, "The square-root unscented kalman filter for state and parameter-estimation," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, vol. 6. IEEE, 2001, pp. 3461–3464.

[20] R. Bellman and R. Kalaba, "On adaptive control processes," *Automatic Control, IRE Transactions on*, vol. 4, no. 2, pp. 1–9, 1959.

[21] A. Feldbaum, "Dual control theory," *Automation and Remote Control*, vol. 21, no. 9, pp. 874–1039, 1960.

[22] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of Artificial Intelligence Research*, vol. 4, pp. 129–145, 1996.

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

[24] J. Kober and J. Peters, "Reinforcement learning in robotics: a survey," in *Reinforcement Learning*. Springer, 2012, pp. 579–610.

[25] N. M. Filatov and H. Unbehauen, *Adaptive dual control: Theory and applications*. Springer, 2004, vol. 302.

[26] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming: An overview," in *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, vol. 1. IEEE, 1995, pp. 560–564.

[27] K. Tan and Y. Li, "Grey-box model identification via evolutionary computing," *Control Engineering Practice*, vol. 10, no. 7, pp. 673 – 684, 2002.

[28] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control," *Automatica*, 2013.

[29] R. Platt Jr, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observations," 2010.

[30] R. He, E. Brunskill, and N. Roy, "Efficient planning under uncertainty with macro-actions," *Journal of Artificial Intelligence Research*, vol. 40, no. 1, pp. 523–570, 2011.

[31] H. Bai, D. Hsu, and W. S. Lee, "Planning how to learn," in *Proc. IEEE Conf. on Robotics and Automation*, 2013.