2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Simultaneous Deep Model-Based Reinforcement Learning and State Inference under Partial Observability

William Cong[1] and Josiah P. Hanna[1]

***Abstract*— Model-based reinforcement learning (MBRL) is a promising approach to enabling robots to learn directly from a limited number of real-world interactions. MBRL is notoriously difficult in settings without full state observability because algorithms must simultaneously infer state from incomplete observations and use these inferences to learn environment dynamics. Toward the use of MBRL for autonomous robots, we introduce EMBRL, an expectation-maximization framework that combines classical Bayesian state estimation with deep MBRL to jointly infer states and learn neural network state transition models. This framework takes advantage of the rich theory and practice of state estimation from the field of robotics, while enabling behavior learning without a priori known robot dynamics. Though conceptually straightforward, our instantiation of this framework for deep MBRL reveals several key challenges when using a learned transition model both for state inference and policy learning. We introduce a practical implementation of EMBRL using both particle and extended Kalman filters and smoothers and discuss key design choices necessary for effective implementation. Finally, we evaluate different instantiations of the EMBRL framework on both simulated and real-robot tasks and show that our methods learn higher performing policies compared to strong MBRL baselines using recurrent neural networks.**

I. Introduction

Humans and other intelligent agents make decisions by reasoning about the consequences of their actions using internal models of their environment. This principle underlies a broad class of methods known as *model-based reinforcement learning* (MBRL), where agents learn a predictive model of the environment's state dynamics and use it to improve their decision-making. While MBRL methods have exhibited great success across a variety of domains, many of these applications assume access to the true state of the environment [1], [2]. However, autonomous robots often operate under *partial observability*, where the true state is hidden and must be inferred from noisy or incomplete observations. This uncertainty complicates both decision-making and model learning. While recurrent models offer a solution by learning a hidden state that summarizes past observations, these methods do not leverage prior knowledge about what factors belong in the estimated state.

Methods for Bayesian state estimation are widely-used in robotics and could, in principle, be used to estimate the unknown states in a robot's experience so that they can then be used for dynamics learning in MBRL. Unfortunately, these methods assume the state dynamics are already known which makes them unsuitable for settings where we want to apply MBRL. While simultaneous expectation-maximization (EM) [3] has been studied for simultaneous model-learning and state inference, prior work has not considered learning deep dynamics models and the integration of these techniques into MBRL. With this gap in mind, our work seeks to answer the following question:

Can we simultaneously learn a deep dynamics model for both MBRL and state inference under partially observability?

To answer this question, we develop a framework for integrating expectation-maximization into MBRL and then identify a set of techniques that enable a practical implementation of the framework using both particle and extended Kalman filters and smoothers. We demonstrate that this combination of techniques enables deep MBRL in partially observable real and simulated learning tasks and leads to improved performance relative to RNN-baselines that do not perform explicit state inference.

II. Related Works

A majority of the POMDP literature focuses on *planning* algorithms which optimize policies in a model of the environment that is assumed to be known a priori [4], [5], [6]. Several methods explore a model-free approach to the full RL problem by augmenting existing model-free algorithms with recurrent neural networks [7], [8], [9], [10]. The EM algorithm has also been used for model-free RL in the context of "policy learning as inference" [11], [12], [13].

In the model-based literature, existing work often assumes full observability (e.g., PILCO, [14]; PETS, [1]; MBPO, [2]; MACURA, [15]). Under partial observability, variational inference techniques that explicitly learn latent representations using recurrent neural networks (e.g. VAEs) are the most common approach (e.g., [16]; PlaNet, [17]; DVRL, [18]) and have been shown to be particularly effective for high-dimensional observation spaces [19]. However, these methods must also learn what information goes in the state representation they construct, which may decrease their sample efficiency in settings where we already know what factors belong in the state. Alternative model-based approaches learn dynamics models with spectral decomposition [20], the Utile Suffix Memory (USM) algorithm [21], Bayesian inference over the model-space ([22]), or rely on the use of Gaussian processes [23], [24].

The combination of Bayesian state estimation and EM for the parameter estimation of state-space models has been extensively explored in the systems literature. Ghahramani and

[1]Computer Sciences Department, University of Wisconsin – Madison, Madison, Wisconsin, U.S.A. wycong@wisc.edu

Hinton [3] derive an analytical solution for the linear-Gaussian setting and Ghahramani and Roweis [25] later extend this approach to the nonlinear setting using Extended Kalman Smoothing (EKS) and radial basis functions. More closely related to this work are [26], [27], [28] which use particle-based methods and a generalized EM procedure, though do not explore using learned models for policy improvement.

III. Background

In this section, we formalize the RL problem under partial observability.

A. Reinforcement Learning

We formalize the environment as a *partially observable Markov decision process* (POMDP) defined by the tuple $\langle S, A, \Omega, T, O, r, d, \gamma \rangle$ where S is the state space, A is the action space, Ω is the observation space, $T : S \times A \times S \to [0, 1]$ is the transition model, $O : S \times A \times \Omega \to [0, 1]$ is the observation model, $r : S \times A \times S \to \mathbb{R}$ is the reward function, $d : S \to [0, 1]$ is the initial state distribution, and $\gamma \in [0, 1]$ is the discount factor.

A POMDP starts in an initial state $s_0 \sim d(\cdot)$. At each time step t, the agent receives an observation $o_t \in \Omega$ generated from the observation model $O(s_t, a_{t-1}, o_t) = p(o_t \mid s_t, a_{t-1})$ and takes an action $a_t \in A$ according to its policy $\pi(a_t \mid b_t)$ where b_t typically represents the agent's belief of the current state s_t. The environment then transitions to the next state s_{t+1} according to the transition model $T(s_t, a_t, s_{t+1}) = p(s_{t+1} \mid s_t, a_t)$ and the agent receives a scalar reward $r_{t+1} = r(s_t, a_t, s_{t+1})$. The goal of *reinforcement learning* (RL) is to learn a policy π^* that maximizes the expected sum of discounted rewards

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \right],$$

where the expectation is taken with respect to trajectories $\tau = (s_0, o_0, a_0, s_1, \ldots)$ induced by the policy π.

B. Nonlinear State-Space Models

We assume that the POMDP is structured as a *discrete-time nonlinear state-space model* (SSM) with transition function f, observation function g, transition noise covariance Q, observation noise covariance R, and initial state distribution $\mathcal{N}(\mu_0, \Sigma_0)$.

$$\begin{aligned} s_t &= f(s_{t-1}, a_{t-1}) + v, \quad v \sim \mathcal{N}(0, Q) \\ o_t &= g(s_t, a_{t-1}) + w, \quad w \sim \mathcal{N}(0, R) \\ s_0 &\sim \mathcal{N}(\mu_0, \Sigma_0) \end{aligned} \tag{1}$$

Transition and observation uncertainty are commonly approximated by static Gaussian noise in the context of robotics and state estimation.

C. Bayesian State Estimation

In many autonomous robotics applications, the state cannot be directly sensed and must instead be estimated based on the sequence of observations made and actions taken. Bayes state estimators (and particularly their practical approximations) provide a computationally tractable approach to estimate the posterior, $p(s_{0:T} \mid o_{0:T}, a_{0:T-1})$, by decomposing estimation into a sequence of one-step recursions that makes inference linear in t.

Bayesian state estimators are categorized by the direction of the recursion. *Filtering* methods compute the marginal posterior of the state s_t in the forward direction using all observations and actions up to the step t. They do so by alternating the following two steps:

$$\begin{aligned} &p(s_t \mid o_{0:t-1}, a_{0:t-1}) && \text{(prediction)} \\ &= \int p(s_t \mid s_{t-1}, a_{t-1})\, p(s_{t-1} \mid o_{0:t-1}, a_{0:t-2})\, ds_{t-1} \\ &p(s_t \mid o_{0:t}, a_{0:t-1}) && \text{(update)} \\ &= \frac{p(o_t \mid s_t, a_{t-1})\, p(s_t \mid o_{0:t-1}, a_{0:t-1})}{\int p(o_t \mid s_t, a_{t-1})\, p(s_t \mid o_{0:t-1}, a_{0:t-1})\, ds_t} \end{aligned}$$

Smoothing methods compute the marginal posterior of the current state s_t in the *backward* direction using all observations and actions up to a future step $T > t$.

$$\begin{aligned} &p(s_t \mid o_{0:T}, a_{0:T-1}) \\ &= p(s_t \mid o_{0:t}, a_{0:t-1}) \\ &\quad \int \frac{p(s_{t+1} \mid s_t, a_t)\, p(s_{t+1} \mid o_{0:T}, a_{0:T-1})}{p(s_{t+1} \mid o_{0:t}, a_{0:t})}\, ds_{t+1} \end{aligned}$$

The equations above define the Bayes-optimal filter and smoother. If $p(s_t \mid s_{t-1}, a_{t-1})$ and $p(o_t \mid s_t, a_{t-1})$ are both known, smoothing will provide a more accurate estimation of the state sequence $s_{0:T}$ as it uses more information in its computed posterior. However, since smoothing estimates $p(s_t \mid o_{0:T}, a_{0:T-1})$ for $t < T$, it can only be used offline when the observations and actions from future time-steps are available. While most settings do not admit closed-form solutions to these updates, the literature on state-estimation contains a vast number of approximations that provide reliable state-estimation in practice. Perhaps two of the most common such approaches are particle and extended Kalman filters and smoothers [29].

IV. EMBRL: A Framework for Deep MRBL Under Partial Observability

In this section, we introduce our framework for MBRL on robots in partially observable settings. At a high-level, our objective is to learn a model of the environment's state transition dynamics, $\hat{p} \approx p(s_{t+1} \mid s_t, a_t)$, and then optimize a policy π with RL using trajectories generated from $\hat{p}$. Accomplishing this objective requires estimating s_t, both to supervise the learning of $\hat{p}$ and to provide π with the state information required for execution on the physical robot. We will use Bayesian state estimation for inferring states; the key challenge is that Bayesian state estimation requires a model of the state transition dynamics for the prediction step which

is precisely the model that we want to estimate in the first place. This results in a chicken-and-egg problem: estimating the transition model requires knowledge of what states were visited, yet state inference requires knowledge of the state transition model.

Our solution is conceptually straightforward: we begin with a randomly initialized state transition model and policy, collect data on the robot, and then alternate between estimating the latent states and then improving the model with these state estimates. This general scheme is an instance of expectation-maximization. While conceptually straightforward, a practical realization with nonlinear function approximation is challenging as both E- and M-steps require approximation and these methods are not inherently robust to significant model error. In the next subsection, we describe a generic EM procedure for our assumed state-space models. Then, in Section V, we describe key techniques that we found enabled its successful implementation in robot control tasks.

Generalized Expectation-Maximization

Our objective for estimating the environment model is to select ϕ to maximize the probability of the robot's observations given its actions, $p_\phi(o_{0:T} \mid a_{0:T-1})$. There are two central challenges that arise with this objective: (1) direct optimization of $p_\phi(o_{0:T} \mid a_{0:T-1})$ is intractable as it requires integrating over the full trajectory of latent states, and (2) filtering and smoothing methods require knowledge of $p(s_t \mid s_{t-1}, a_{t-1})$ and $p(o_t \mid s_t, a_{t-1})$. The latter challenge is the aforementioned chicken-and-egg problem of model estimation in POMDPs.

The *expectation-maximization* (EM) algorithm ([30]) provides a solution to both of these problems by alternating between two steps: In the E-step, the model parameters ϕ are fixed and the marginal state posteriors $q_\phi(s_{0:T} \mid o_{0:T}, a_{0:T-1})$ are computed using the filtering and smoothing methods discussed in Section III-C. In the M-step, the model parameters are updated to maximize the expected complete-data log-likelihood, given by

$$\mathcal{Q}(\phi; q_{\phi^{(\text{old})}}) \coloneqq \mathbb{E}_{q_{\phi^{(\text{old})}}}\left[\log p_\phi(o_{0:T}, s_{0:T} \mid a_{0:T-1})\right], \quad (2)$$

where the expectation is taken with respect to the posteriors computed in the E-step. EM avoids direct optimization of the marginal likelihood through the surrogate objective $\mathcal{Q}(\phi; q_{\phi^{(\text{old})}})$. By a simple application of Jensen's inequality and the introduction of a variational distribution $q(s_{0:T}) \coloneqq q_{\phi^{(\text{old})}}(s_{0:T} \mid o_{0:T}, a_{0:T-1})$, we can see that the maximization of $\mathcal{Q}(\phi; q_{\phi^{(\text{old})}})$ improves a lower bound on $\log p_\phi(o_{0:T} \mid a_{0:T-1})$:

$$\begin{aligned}
&\log p_\phi(o_{0:T} \mid a_{0:T-1}) \\
&= \log \int p_\phi(o_{0:T}, s_{0:T} \mid a_{0:T-1})\, ds_{0:T} \\
&\geq \int q(s_{0:T}) \log \frac{p_\phi(o_{0:T}, s_{0:T} \mid a_{0:T-1})}{q(s_{0:T})}\, ds_{0:T} \\
&= \mathcal{Q}(\phi; q_{\phi^{(\text{old})}}) + \mathcal{H}(q_{\phi^{(\text{old})}}(s_{0:T} \mid o_{0:T}, a_{0:T-1}))
\end{aligned}$$

Importantly, the structural assumptions of the state-space model ensure that the joint log-likelihood $\log p_\phi(o_{0:T}, s_{0:T} \mid a_{0:T-1})$ can be evaluated in closed-form. Under the linear-Gaussian assumption, there exists an exact analytical solution for the M-step [31]. We extend this approach to the general nonlinear setting by replacing the learned components of the state-space model with neural networks and optimizing $\mathcal{L}(\phi) = -\mathcal{Q}(\phi; q_{\phi^{(\text{old})}})$ with gradient descent. Plugging in the state-space model assumptions (1), the EM objective is given by:

$$\mathcal{L}(\phi) = -\mathbb{E}_{q_{\phi^{(\text{old})}}(s_{0:T} \mid o_{0:T}, a_{0:T-1})}[\ell_\phi(s_{0:T}, o_{0:T}, a_{0:T-1})], \quad (3)$$

$$\begin{aligned}
\ell_\phi(s_{0:T}, o_{0:T}, a_{0:T-1}) = &\frac{1}{2}\left[(\Delta_0^s)^\top \Sigma_0^{-1} \Delta_0^s + \log|\Sigma_0|\right] \\
&+ \frac{1}{2}\left[\sum_{t=0}^{T} (\Delta_t^o)^\top R^{-1} \Delta_t^o + \log|R|\right] \\
&+ \frac{1}{2}\left[\sum_{t=1}^{T} (\Delta_t^s)^\top Q^{-1} \Delta_t^s + \log|Q|\right] \quad (4)
\end{aligned}$$

where $\Delta_0^s = s_0 - \mu_0$, $\Delta_t^o = o_t - g(s_t, a_{t-1})$, and $\Delta_t^s = s_t - f(s_{t-1}, a_{t-1})$. In theory, this objective can be optimized to learn any subset of $\{\mu_0, \Sigma_0, f, Q, g, R\}$. However, jointly estimating all or many of these components may be non-identifiable without additional assumptions or constraints as multiple parameter settings can induce the same distribution over observed trajectories. With this in mind, we focus on learning the transition dynamics $\{f, Q\}$ and assume $\{\mu_0, \Sigma_0, g, R\}$ are known.

V. Implementation: General Mechanisms

In this section, we provide a general outline of the EMBRL framework in Algorithm 1 and discuss algorithmic choices that are filter- and smoother-agnostic. Their practical realization is examined in Section VI.

A. Decoupling Transition Noise for State Estimation and Learning

State estimation algorithms generally assume access to an accurate transition model. If the transition model is misspecified—as it invariably is during model learning—we must increase the transition noise used during state estimation to account for this additional error. Intuitively, the transition noise used in a filter's prediction step reflects both stochasticity in state transitions and uncertainty over the next state. If the state estimation algorithm is overly confident in the predictions of a misspecified transition model, it will discount observations and drift away from the true latent trajectory.

To mitigate this problem, we maintain a separate, artificially inflated transition noise $\hat{Q}$ that is used exclusively for state estimation. In our implementation, we simply add a fixed diagonal offset to the learned Q_ϕ. Alternatively, Q_ϕ can be scaled by a factor that is gradually annealed as the transition model improves.

B. Evidence-Gated Optimization

When the E- and M-steps are exact, each iteration of the EM algorithm is guaranteed to monotonically improve the marginal log-likelihood $\log p_\phi(o_{0:T} \mid a_{0:T-1})$. However, our

method relies on approximate state estimation techniques in the E-step and stochastic gradient updates in the M-step which break the monotonicity guarantee and may lead to parameter updates that decrease the objective.

To mitigate this issue, we ensure that parameter updates are accepted only when they improve the variational bound. Concretely, after performing an E-step on a batch of trajectories, we gate the following M-step by comparing the evidence lower bound (ELBO) of the new posteriors and parameters $\{q',\ \phi'\}$ against the ELBO of the previous posteriors and parameters $\{q,\ \phi\}$. That is, we only accept the updated ϕ' if:

$$\mathcal{F}(q', \phi') - \mathcal{F}(q, \phi) \geq 0, \quad \mathcal{F}(q, \phi) \coloneqq \mathcal{Q}(q, \phi) + \mathcal{H}(q)$$

For successive M-steps with respect to a fixed posterior q, the entropy is constant. Thus, it is sufficient to gate with respect to the EM objective:

$$\mathcal{Q}(q, \phi') - \mathcal{Q}(q, \phi) \geq 0$$

If an M-step directly following an E-step is rejected, we skip to the next batch. If the parameters have been updated since the last E-step, we continue to the next E-step. Note that the first EM iteration on each batch is always performed.

While we choose to gate at every iteration, this is computationally expensive and often unnecessary. One could consider restricting gating decisions to a subset of iterations or to apply them only during the early phases of learning, when model error and posterior variance are largest. Moreover, Monte Carlo estimates of $\mathcal{Q}$ and $\mathcal{H}$ are subject to estimator variance, which may lead to spurious rejections. To mitigate this issue, one can introduce a tolerance threshold, accepting updates as long as the estimated improvement exceeds $-\epsilon$ where ϵ is chosen in proportion to the estimator variance. In domains with near-linear dynamics, low noise, or well-specified models, we find that gating may be omitted entirely.

C. Evidence-Gated Smoothing

Although smoothing leverages information from the full observation sequence, the quality of its estimated posteriors is more sensitive to a misspecified transition model as errors in the dynamics will be propagated through the latent trajectory a second time.

To address this issue, we introduce a per-trajectory selection process in which we compute the ELBO under both the filtered and smoothed posteriors and choose whichever is higher for the subsequent M-step(s). In practice, performing this selection process after every E-step can be computationally expensive and unnecessary. Similar to Section V-B, a more practical approach is to restrict this selection process to a subset of iterations or to the early phases of training before switching to the smoothed posteriors exclusively once the transition model has improved.

D. Model Architecture

We parameterize the transition model $f_\phi(s, a)$ with a feed-forward neural network that predicts the expected change in state rather than the next state i.e., $s' = f_\phi(s, a) + s$. This approach eliminates the need to memorize the input state and has been shown to accelerate learning and improve generalization in continuous control domains ([32]). Since Q_ϕ is a static quantity, we directly parameterize its Cholesky factors to ensure positive definiteness throughout training.

E. Policy Optimization & Execution

After each model learning step, we train the policy using *proximal policy optimization* (PPO) ([33]) in the learned MDP with transition dynamics $p(s' \mid s, a) = \mathcal{N}(s + f_\phi(s, a), Q_\phi)$. PPO is a model-free policy gradient algorithm that encourages conservative optimization of the policy by clipping the policy and value function updates.

Since the learned policy is conditioned on the latent states, executing the policy in the real-world requires simultaneously executing the filter to obtain the agent's current belief of the environment state, $b_t(s) = p(s_t \mid o_{0:t}, a_{0:t-1})$. If the belief is unimodal, conditioning the policy on the posterior mean $\mu_t = \mathbb{E}[s_t \mid o_{0:t}, a_{0:t-1}]$ is often sufficient. When the policy is multi-modal, the policy may require a richer summary of the belief distribution.

Algorithm 1 EMBRL (for learning $\{f, Q\}$)

Initialize policy π_θ and replay buffer D.
Initialize f_ϕ and Q_ϕ according to Section V-D.
for each outer step **do**
  Sample trajectories $\tau \sim \pi_\theta$ where π_θ is conditioned on beliefs obtained by the filter.
  $D \cup \{\tau\}$
  for each model learning step **do**
    Sample a batch of n trajectories from D.
    for each EM step **do**
      Compute filtered posteriors.
      Compute smoothed posteriors.
      Select posteriors according to Section V-C).
      for each M step **do**
        Optimize according to Section V-B).
      end for
    end for
    Optimize π_θ using PPO in the MDP with initial state distribution $d(s_0) = \mathcal{N}(\mu_0, \Sigma_0)$ and transition dynamics $p(s' \mid s, a) = \mathcal{N}(s + f_\phi(s, a), Q_\phi)$.
  end for
end for

F. Prior-Assisted State Estimation

The explicit separation between state estimation and model optimization enables the use of different models for inference and learning. In robotics domains, it is common to have access to an approximate or partial transition model $\tilde{f}$ (e.g., a motion model for localization). This knowledge can be integrated into the learning process by performing E-steps in place of the learned model f_ϕ during the early stages of training. Under significant partial observability, learning directly from a randomly initialized model may be infeasible. By injecting informative state estimates from $\tilde{f}$, the M-step can make meaningful progress early on and progressively refine $\tilde{f}$ using the observed trajectories.

VI. Implementation: State Estimation

Next, we discuss two practical implementations of the state estimation step. Since the filtering and smoothing algorithms discussed in this section are well-known, we omit their technical details and instead focus on their role within EM. In particular, we examine how $\mathcal{Q}$ and $\mathcal{H}$ can be computed with both the filtered and smoothed posteriors as this is required for implementation of the mechanisms discussed in Section V.

A. EK-EMBRL

We present an implementation of the EMBRL framework using an extended Kalman filter (EKF) and extended Kalman smoother (EKS). The EKF and EKS produce Gaussian posteriors which enable a closed-form expression for $\mathcal{Q}$ and $\mathcal{H}$.

Computing $\mathcal{Q}$: Given the smoothed statistics $\{m_t^s, P_t^s, P_{t,t-1}^s\}_{t=0}^T$ where $m_t^s = \mathbb{E}[s_t \mid o_{0:T}, a_{0:T-1}]$, $P_t^s = \mathrm{Cov}(s_t \mid o_{0:T}, a_{0:T-1})$, and $P_{t,t-1}^s = \mathrm{Cov}(s_t, s_{t-1} \mid o_{0:T}, a_{0:T-1})$, $\mathcal{Q}$ can be computed as

$$\mathcal{Q}(\phi; q_{\phi^{(\text{old})}}) = \\ -\frac{1}{2}\left[\sum_{t=1}^{T}(\Delta_t)^\top Q_\phi^{-1}\Delta_t + \mathrm{tr}(Q_\phi^{-1}\Xi_t) + \log|2\pi Q_\phi|\right] + \text{const}$$

where $\Delta_t = m_t^s - f_\phi(m_{t-1}^s, a_{t-1})$ and $\Xi_t = P_t^s + F_t P_{t-1}^s F_t^\top - F_t(P_{t,t-1}^s)^\top - P_{t,t-1}^s F_t^\top$. F_t and G_t denote the Jacobians of the transition and observation models respectively and are evaluated at the corresponding smoothed means. $\mathcal{Q}$ can be computed from the filtered statistics by replacing $\{m_t^s, P_t^s, P_{t,t-1}^s\}_{t=0}^T$ with $\{m_t^f, P_t^f, P_{t,t-1}^f\}_{t=0}^T$ where $P_{t,t-1}^f$ denotes the filtered lag-one covariance given by $P_{t,t-1}^f = (I - K_t G_t) F_t P_{t-1}$. In this case, F_t and G_t are evaluated at the filtered means.

Computing $\mathcal{H}$: The entropy of the smoothed posteriors can be computed exactly and is given by

$$\mathcal{H}(q) = \frac{1}{2}\log((2\pi e)^d|P_0^s|) + \frac{1}{2}\sum_{t=1}^{T}\log((2\pi e)^d|\Sigma_{t,t-1}^s|),$$

where $\Sigma_{t,t-1}^s = P_t^s - P_{t,t-1}^s(P_{t-1}^s)^{-1}(P_{t,t-1}^s)^\top$. The entropy of the filtered posteriors can be obtained by replacing $\{P_t^s, P_{t,t-1}^s\}_{t=0}^T$ with $\{P_t^f, P_{t,t-1}^f\}_{t=0}^T$.

B. P-EMBRL

Next, we present a non-parametric alternative that uses a bootstrap particle filter (BPF) and a backward-simulation particle smoother (BSPS). Particle-based methods offer greater flexibility compared to the Gaussian assumption of the EKF/EKS and are a natural solution for handling strong nonlinearities and multimodality.

Computing $\mathcal{Q}$: The BSPS samples trajectories from an approximation of the full posterior as part of the smoothing process—these samples can be directly used to compute a Monte Carlo approximation of $\mathcal{Q}$:

$$\mathcal{Q}_{\text{MC}}(\phi; q_{\phi^{(\text{old})}}) = \frac{1}{N}\sum_{i=1}^{N}\ell_\phi(s_{0:T}^{(i)}, o_{0:T}, a_{0:T-1}),$$
$$s_{0:T}^{(i)} \sim q_{\phi^{(\text{old})}}(s_{0:T} \mid o_{0:T}, a_{0:T-1}) \text{ for } i = 1, \ldots, N$$

Without a smoother, complete trajectories can be reconstructed by sampling a terminal particle and tracing back through its ancestors. These ancestor-sampled paths approximate the smoothing distribution, and the same above estimator applies.

Computing $\mathcal{H}$: The entropy of the filtered and smoothed posteriors can be approximated using the plug-in Gaussian entropy

$$\mathcal{H}(q) = \frac{1}{2}\sum_{t=0}^{T}\log\big((2\pi e)^d|\Sigma_t|\big),$$

where Σ_t is the marginal sample covariance of the filtered or smoothed particles at time t. Given forward particles $\{z_t^{(i)}\}_{i=1}^N$ and corresponding weights $\{w_t^{(i)}\}_{i=1}^N$, the filtered marginal covariances can be computed as $\Sigma_t = \sum_{i=1}^N w_t^{(i)}\big(z_t^{(i)} - \mu_t\big)\big(z_t^{(i)} - \mu_t\big)^\top$ where $\mu_t = \sum_{i=1}^N w_t^{(i)} z_t^{(i)}$. Given smoothed trajectories $\tau^{(m)} = (z_{0:T}^{(m)})$, the smoothed marginal covariances can be computed as $\Sigma_t = (1/M)\sum_{m=1}^M \big(z_t^{(m)} - \mu_t\big)\big(z_t^{(m)} - \mu_t\big)^\top$ where $\mu_t = (1/M)\sum_{m=1}^M z_t^{(m)}$. While the plug-in Gaussian approximation to the posterior entropy is admittedly crude, we only use the entropy for gating updates and gating requires only relative comparisons across candidates, for which this surrogate is often sufficient in practice.

VII. Experimental Analysis

We evaluate instantiations of the EMBRL framework on a set of simulated and real-world control tasks.

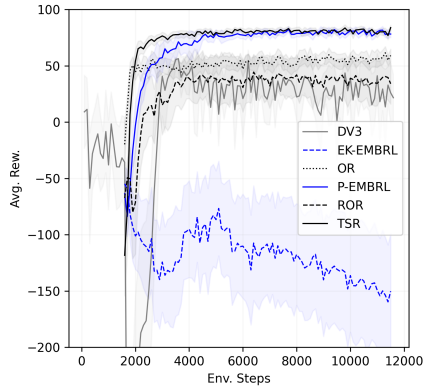
A. Empirical Setup

We compare EMBRL methods against several model learning techniques: (1) observation regression (OR), which trains a feed-forward dynamics model directly on the observations by minimizing the one-step prediction MSE; (2) recurrent observation regression (ROR), which replaces the feed-forward model in OR with an LSTM; and (3) true state regression (TSR), which trains a feed-forward dynamics model on the one-step prediction MSE with respect to the true states. We also compare against DreamerV3 [19], a state-of-the-art model-based reinforcement learning algorithm that serves as a strong benchmark for sample-efficient MBRL.

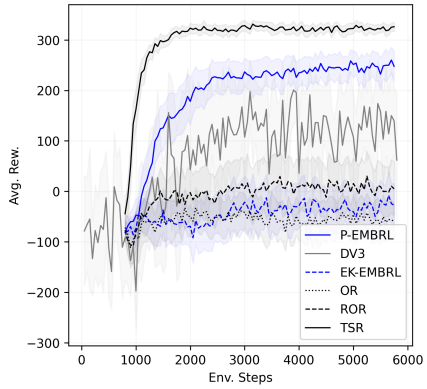
The learned policies for OR and ROR are executed directly in the true POMDP. TSR requires executing a filter to obtain a latent belief—Figures 1a-1e use a BPF, Figure 1f uses an EKF. Exploration during data collection is performed by perturbing the actions with Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.2I)$. Apart from the differences in model learning and policy execution, all other components of OR, ROR, and TSR (i.e., the model architecture, hyperparameters) are identical to that of EK-EMBRL and P-EMBRL. This ensures that any differences in performance can be attributed to the core design choices under evaluation.

B. Simulation Results

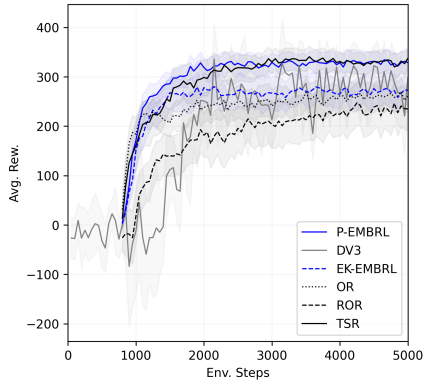
Here we analyze our simulation results as shown in Figures 1a-1f. In Figures 1a-1e, the transition dynamics are learned from scratch. We observe that P-EMBRL produces an



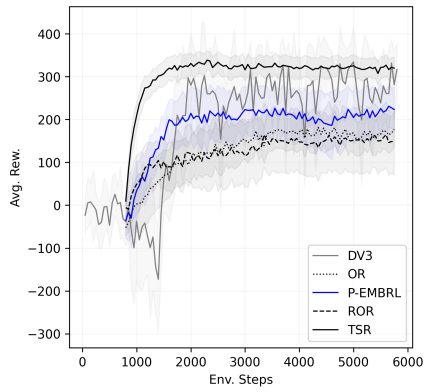
(a) The classical inverted pendulum task where the agent is rewarded for balancing the pendulum in the upright position. The observation model directly outputs the latent state but introduces significant observation noise ($R = I$).



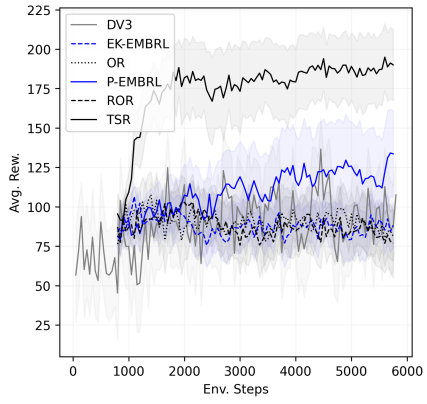
(b) A 2D navigation task with highly nonlinear dynamics. The observation model provides near-identity measurements of position and heading, perturbed by sinusoidal distortions and significant observation noise ($R = I$).



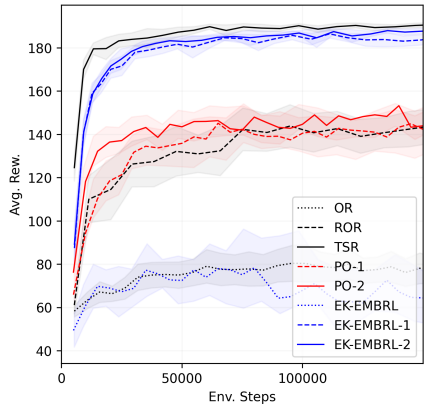
(c) A 2D navigation task with standard differential-drive kinematics. The observation model provides direct access to the latent state in addition to a small amount of noise ($R = 0.1I$).



(d) A 2D navigation task with differential-drive dynamics. The observation model provides 16 simulated LiDAR range measurements perturbed by significant observation noise ($R = I$). We do not evaluate EK-EMBRL on this task as the observation model is discontinuous.



(e) A 2D two-link robotic arm with frictional joint dynamics. The agent is rewarded for moving the end-effector to a target location. The observation model provides measurements of the joint angles perturbed by a small amount of noise ($R = 0.1I$).



(f) An occluded inverted pendulum system in which the velocities of the base and pendulum are hidden from the agent. We assume the transition noise is known and that both transition and observation noise are low ($Q = 10^{-6}I$, $R = 10^{-4}I$).

Fig. 1: Simulation results. We run 30 trials for each method and report the 95% confidence interval computed using the Student's t-distribution. All domains apart from Figure 1d assume access to a known reward function. In Figure 1d, OR, ROR, TSR, and P-EMBRL learn a reward model by minimizing one-step predictions. The reward model is trained on the observations for OR and ROR, the true states for TSR, and the mean of the latent posteriors in P-EMBRL.

average reward higher than all baselines except the privileged TSR in four of the five tasks, performing slightly worse than DreamerV3 in Figure 1d. As expected, the particle-based E-step excels in noisy, highly nonlinear domains. By contrast, EK-EMBRL performs poorly in these settings but attains acceptable performance in a low-noise environment (Figure 1c).

Figure 1f demonstrates how a prior model can be used to assist state estimation in a setting where inference is especially challenging. We evaluate a variant of the EK-EMBRL algorithm on an occluded inverted pendulum system in which the agent only observes the position of the base x and angle of the pendulum θ. We assume access to two misspecified models of the pendulum's dynamics obtained by perturbing the parameters of the true transition model f.

Parameter	f	$\tilde{f}_1$	$\tilde{f}_2$
Base Mass (kg)	1.0	1.0	4.0
Pendulum Mass (kg)	1.0	1.0	3.0
Pendulum Length (m)	1.0	5.0	2.0
Force Coef. (N)	10.0	3.0	10.0

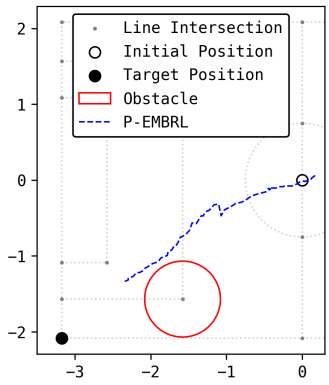
In addition to the OR, ROR, and TSR baselines described above, we evaluate the following methods: (1) EK-EMBRL, which learns the transition model from scratch, (2) EK-EMBRL-1 and EK-EMBRL-2, which are identical to EK-EMBRL except the first E-step on each batch is performed using the perturbed transition models $\tilde{f}_1$ and $\tilde{f}_2$, and (3) PO-1 and PO-2, which learn a policy directly in the MDP defined by $\tilde{f}_1$ and $\tilde{f}_2$. In all of the methods described above, the learned policy is executed with an EKF consistent with the

transition model used during training. The reward function is assumed to be a known function of x and θ.

In this setting, EK-EMBRL fails to produce a strong policy when learning the dynamics from scratch and performs on par with OR. While the perturbed models are similarly incapable of producing a good policy in isolation, Figure 1f shows that $\tilde{f}_1$ and $\tilde{f}_2$ can be used as a good starting point for EK-EMBRL to yield a policy that approaches TSR.

C. Real-World Results

We evaluate our approach on a real-world navigation task using the NAOv6 humanoid robot. The task requires the robot to walk to a target location on the soccer field while avoiding a fixed obstacle along its path. At each step, the robot observes the relative position of the field line intersections within its field of view.



(a) A visualization of the navigation task and the landmarks that can be observed. The dashed blue line depicts the trajectory of a policy learned using the P-EMBRL algorithm.

(b) The actual NAOv6 robot and soccer field used in the experiment. The robot perceives its environment through two head-mounted cameras, providing a field of view of $54.7°$.

We compare a variant of the P-EMBRL algorithm against the OR, ROR and TSR baselines. Because the correspondence between observed landmarks and true field intersections is unknown, we perform maximum likelihood data association using the Hungarian algorithm in the BPF update step. For TSR, we learn the transition model on the estimated states obtained from the robot's on-board localization module (which incorporates more information than just field line intersections and consequently provides accurate state estimates).

Each method is trained on a dataset containing $n = 50$ trajectories of length $T = 60$. All trajectories are collected using a fixed forward-walking policy that is perturbed with pink action noise and executed at 5 hz. This domain is particularly challenging as the raw NAO observations are noisy and unreliable. To accelerate learning, we pre-process the data by aligning each observation with the position of its corresponding landmark using the robot's localized states.

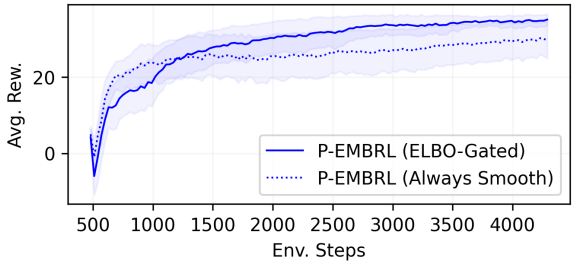
We report the total reward obtained by each method in Table 3. Across all datasets, P-EMBRL and TSR significantly outperform OR and ROR. This result demonstrates that P-EMBRL can perform nearly as well as a MBRL baseline with access to accurate state estimates in a setting where learning a hidden state representation with an RNN for MBRL fails.

Method	Mean ± 95% CI	Median	Min	Max
P-EMBRL	2.46 ± 0.20	2.65	1.29	3.00
OR	-0.05 ± 0.14	-0.09	-0.64	0.55
ROR	0.17 ± 0.16	0.27	-0.94	0.76
TSR	**2.61 ± 0.24**	2.87	1.10	3.44

Fig. 3: Summary statistics for the total reward obtained by each method across all 5 datasets and 5 runs per dataset. The rewards are computed using the estimated states from on-board localization.

D. Ablation: Evidence-Gated Smoothing

Finally, we demonstrate the benefit of evidence-gated smoothing in a simulation of the setting described in Section VII-C. We compare the data-association variant of P-EMBRL that always optimizes the smoothed posteriors against one that selectively optimizes whichever of the filtered or smoothed posteriors achieves a higher ELBO, using the approximation outlined in Section VI.



We observe that the ELBO-gated variant consistently produces a policy that obtains higher reward. By gating on the ELBO, we avoid reinforcing poor associations while still retaining the benefits of smoothing when the associations are correct.

VIII. Conclusion

In this paper, we introduced EMBRL, an EM framework for deep MBRL that uses Bayesian state estimation to learn dynamics models under partial observability. We presented a practical implementation using particle-based filtering and smoothing and discussed key design choices to guide implementation. Finally, we evaluated several instantiations of the EMBRL framework on a set of continuous control tasks and found that it enabled learning models which supported effective policy optimization in partially observable tasks.

Limitations & future work: We identify key limitations of the current approach and outline directions for future work. First, it is difficult to disentangle transition noise from model error when jointly learning f and Q. In practice, the learned transition noise will inflate to compensate for errors in f which results in an overestimation of the true transition noise. A deeper understanding of the identifiability of $\{\mu_0, \Sigma_0, f, Q, g, R\}$ in the context of MBRL could provide stronger theoretical guarantees and better inform algorithmic design. Two additional limitations of EMBRL are its reliance on (1) naive action-noise exploration and (2) model rollouts

from a known initial state distribution. Future work should explore uncertainty-driven exploration techniques as well as branched rollout strategies (e.g. [2]) to mitigate compounding model error. Lastly, the assumption of time-invariant Gaussian noise may be overly simplistic. Extending the EMBRL framework to handle time-varying, heteroscedastic, or non-Gaussian noise could broaden its applicability to more complex and realistic settings.

ACKNOWLEDGMENT

This work took place in the Prediction and Action Lab (PAL) at the University of Wisconsin – Madison. PAL research is supported by NSF (IIS-2410981) and the Wisconsin Alumni Research Foundation.

REFERENCES

[1] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," 2018. [Online]. Available: https://arxiv.org/abs/1805.12114

[2] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," 2021. [Online]. Available: https://arxiv.org/abs/1906.08253

[3] Z. Ghahramani and G. E. Hinton, "Parameter estimation for linear dynamical systems," 1996. [Online]. Available: https://api.semanticscholar.org/CorpusID:12534912

[4] A. G. Barto, S. J. Bradtke, and S. P. Singh, "Learning to act using real-time dynamic programming," *Artificial Intelligence*, vol. 72, no. 1, pp. 81–138, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/000437029400011O

[5] M. T. Spaan and N. Vlassis, "Perseus: Randomized point-based value iteration for pomdps," *Journal of Artificial Intelligence Research*, vol. 24, p. 195–220, Aug. 2005. [Online]. Available: http://dx.doi.org/10.1613/jair.1659

[6] J. M. Porta, N. Vlassis, M. T. Spaan, and P. Poupart, "Point-based value iteration for continuous pomdps," *Journal of Machine Learning Research*, vol. 7, no. 83, pp. 2329–2367, 2006. [Online]. Available: http://jmlr.org/papers/v7/porta06a.html

[7] B. Bakker, "Reinforcement learning with long short-term memory," in *Advances in Neural Information Processing Systems*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds., vol. 14. MIT Press, 2001. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2001/file/a38b16173474ba8b1a95bcbc30d3b8a5-Paper.pdf

[8] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," 2017. [Online]. Available: https://arxiv.org/abs/1507.06527

[9] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate to solve riddles with deep distributed recurrent q-networks," 2016. [Online]. Available: https://arxiv.org/abs/1602.02672

[10] P. Zhu, X. Li, P. Poupart, and G. Miao, "On improving deep reinforcement learning for pomdps," 2018. [Online]. Available: https://arxiv.org/abs/1704.07978

[11] M. Toussaint, S. Harmeling, and A. Storkey, "Probabilistic inference for solving (po)mdp's," Dec 2006. [Online]. Available: http://www.anc.ed.ac.uk/~amos/publications/ToussaintetAl2006ProbabilisticInferenceForSolvingPOMDPs.pdf

[12] J. Pajarinen and J. Peltonen, "Expectation maximization for average reward decentralized pomdps," in *Machine Learning and Knowledge Discovery in Databases*, H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 129–144.

[13] M. Liu, X. Liao, and L. Carin, "Online expectation maximization for reinforcement learning in pomdps," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, ser. IJCAI '13. AAAI Press, 2013, p. 1501–1507.

[14] M. P. Deisenroth and C. E. Rasmussen, "Pilco: a model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11. Madison, WI, USA: Omnipress, 2011, p. 465–472.

[15] B. Frauenknecht, A. Eisele, D. Subhasish, F. Solowjow, and S. Trimpe, "Trust the model where it trusts itself – model-based actor-critic with uncertainty-aware rollout adaption," 2024. [Online]. Available: https://arxiv.org/abs/2405.19014

[16] D. Ha and J. Schmidhuber, "World models," 2018. [Online]. Available: https://zenodo.org/record/1207631

[17] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," 2019. [Online]. Available: https://arxiv.org/abs/1811.04551

[18] M. Igl, L. Zintgraf, T. A. Le, F. Wood, and S. Whiteson, "Deep variational reinforcement learning for pomdps," 2018. [Online]. Available: https://arxiv.org/abs/1806.02426

[19] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering diverse domains through world models," 2024. [Online]. Available: https://arxiv.org/abs/2301.04104

[20] K. Azizzadenesheli, A. Lazaric, and A. Anandkumar, "Reinforcement learning of pomdps using spectral methods," 2016. [Online]. Available: https://arxiv.org/abs/1602.07764

[21] G. Shani, R. I. Brafman, and S. E. Shimony, "Model-based online learning of pomdps," in *Machine Learning: ECML 2005*, J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, and L. Torgo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 353–364.

[22] S. Ross, B. Chaib-draa, and J. Pineau, "Bayesian reinforcement learning in continuous pomdps with application to robot navigation," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 2845–2851.

[23] P. Dallaire, C. Besse, S. Ross, and B. Chaib-draa, "Bayesian reinforcement learning in continuous pomdps with gaussian processes," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 2604–2609.

[24] M. Deisenroth and J. Peters, "Solving nonlinear continuous state-action-observation pomdps for mechanical systems with gaussian noise," 2012.

[25] Z. Ghahramani and S. Roweis, "Learning nonlinear dynamical systems using an em algorithm," in *Advances in Neural Information Processing Systems*, M. Kearns, S. Solla, and D. Cohn, Eds., vol. 11. MIT Press, 1998. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1998/file/0ebcc77dc72360d0eb8e9504c78d38bd-Paper.pdf

[26] J. Kokkala, A. Solin, and S. Särkkä, "Expectation maximization based parameter estimation by sigma-point and particle smoothing," in *17th International Conference on Information Fusion (FUSION)*, 2014, pp. 1–8.

[27] R. Frigola, F. Lindsten, T. B. Schön, and C. E. Rasmussen, "Identification of gaussian process state-space models with particle stochastic approximation em," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 4097–4102, 2014, 19th IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667016422445

[28] N. Kantas, A. Doucet, S. S. Singh, J. Maciejowski, and N. Chopin, "On particle methods for parameter estimation in state-space models," *Statistical Science*, vol. 30, no. 3, Aug. 2015. [Online]. Available: http://dx.doi.org/10.1214/14-STS511

[29] D. F. Sebastian Thrun, Wolfram Burgard, *Probabilistic Robotics*. MIT Press, 2005.

[30] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977. [Online]. Available: http://www.jstor.org/stable/2984875

[31] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[32] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," 2017. [Online]. Available: https://arxiv.org/abs/1708.02596

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347