DualSMC: Tunneling Differentiable Filtering and Planning under Continuous POMDPs

Yunbo Wang[1*], **Bo Liu**[2*], **Jiajun Wu**[3], **Yuke Zhu**[2], **Simon S. Du**[4], **Li Fei-Fei**[3] and **Joshua B. Tenenbaum**[5]
[1]Tsinghua University
[2]The University of Texas at Austin
[3]Stanford University
[4]Institute for Advanced Study
[5]Massachusetts Institute of Technology
yunbo.thu@gmail.com, bliu@cs.utexas.edu

Abstract

A major difficulty of solving continuous POMDPs is to infer the multi-modal distribution of the unobserved true states and to make the planning algorithm dependent on the perceived uncertainty. We cast POMDP filtering and planning problems as two closely related Sequential Monte Carlo (SMC) processes, one over the real states and the other over the future optimal trajectories, and combine the merits of these two parts in a new model named the DualSMC network. In particular, we first introduce an *adversarial particle filter* that leverages the adversarial relationship between its internal components. Based on the filtering results, we then propose a planning algorithm that extends the previous SMC planning approach [Piche *et al.*, 2018] to continuous POMDPs with an uncertainty-dependent policy. Crucially, not only can DualSMC handle complex observations such as image input but also it remains highly interpretable. It is shown to be effective in three continuous POMDP domains: the floor positioning domain, the 3D light-dark navigation domain, and a modified Reacher domain[†].

1 Introduction

Partially Observable Markov Decision Processes (POMDPs) formulate reinforcement learning problems where the agent's instant observation is insufficient for optimal decision making [Kaelbling *et al.*, 1998]. Decision making with partial observations requires taking the history into account, which brings a high computation cost. It is a known result that finding the optimal policy in finite-horizon POMDPs is PSPACE-complete [Papadimitriou and Tsitsiklis, 1987], which makes POMDPs difficult to solve in moderately large discrete spaces, let alone *continuous domains*.

Approximate solutions to POMDPs based on deep reinforcement learning can directly encode the history of past observations with deep models like RNNs [Hausknecht and Stone, 2015; Karkus *et al.*, 2017; Zhu *et al.*, 2018; Igl *et al.*, 2018; Hafner *et al.*, 2019]. Learning is done in an end-to-end fashion and the resulting models can handle complex observations including visual inputs. However, since conventional POMDP problems usually present an *explicit state formulation*, executing the planning algorithm in a latent space makes it difficult to adopt any useful prior knowledge. Besides, whenever these models fail to perform well, it is difficult to analyze which part causes the failure as they are less interpretable.

In this work, we present a simple but effective model named Dual Sequential Monte Carlo (DualSMC). It preserves high interpretability since the state belief is represented by particles in real state spaces. It is also flexible for solving continuous POMDPs with complex observations and unknown dynamics. The idea of DualSMC is inspired by the recent successes on differentiable particle filters [Jonschkowski *et al.*, 2018; Karkus *et al.*, 2018] and the *control as inference* framework [Kappen *et al.*, 2012; Levine, 2018; Piche *et al.*, 2018]. In particular, DualSMC solves continuous POMDPs by connecting a SMC filter for state estimation with a SMC planner that samples in the optimal future trajectory space[*].

Since the performance of the planner significantly depends on that of the filter, we introduce a novel adversarial training method to enhance the filter. Moreover, to connect the two parts and reduce the computational burden, we feed the top candidates of the state particles into the planner as the initial belief for uncertainty-aware action selection. The planner also takes as input the mean of the top state particles. To further improve robustness, we perform the *model predictive control* where only the first action of the plan is selected and we replan at each step. Notably, the learned dynamics is efficiently *shared* between filtering and model-based planning.

Our contributions to continuous POMDPs with DualSMC can be summarized as follows:

- It proposes a new *differentiable particle filter* (DPF) that leverages the adversarial relationship between the internals of the original DPF [Jonschkowski *et al.*, 2018].

[*]Equal contribution

[†]Code available at https://github.com/Cranial-XIX/DualSMC

[*]To distinguish the two SMCs, we call the first state estimation SMC the *filter* and its particles the *state particles*, and the second planning SMC the *planner* and its particles the *trajectory particles*.

- It introduces a new POMDP planning algorithm in forms of neural networks that extends the original sequential Monte Carlo planning [Piche *et al.*, 2018] from fully observable scenarios to partially observable ones. The algorithm ties the transition model between the filter and the planner, bridges them via particle-based belief states, and learns the uncertainty-aware policy from the belief.
- It provides new benchmarks for continuous POMDPs: the *floor positioning* for explanatory purposes, the 3D *light-dark* navigation with rich visual inputs, and a control task in the Mujoco environment [Todorov *et al.*, 2012]. DualSMC achieves the best results consistently.

2 Problem Setup

A continuous POMDP can be usually specified as a 7-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{Z}, \gamma)$, where $\mathcal{S}$, $\mathcal{A}$ and Ω are continuous state, action and observation spaces. We denote $s_t \in \mathcal{S}$ as the underlying state at time t. When the agent takes an action $a_t \in \mathcal{A}$ according to a policy $\pi(a_t|o_{\leq t}, a_{<t})$, the state changes to s_{t+1} with probability $\mathcal{T}(s_{t+1}|s_t, a_t)$. The agent will then receive a new observation $o_{t+1} \sim \mathcal{Z}(o_{t+1}|s_{t+1})$ and a reward $r_t \sim \mathcal{R}(s_t, a_t)$. Assuming the episodes are of fixed length L, the agent's objective is then to maximize the expected cumulative future reward $G = \mathbb{E}_{\tau\sim\pi}\big[\sum_{t=1}^{L} \gamma^{t-1} r_t\big]$, where $\tau = (s_1, a_1, \ldots, a_L, s_{L+1})$ are trajectories induced by π, and $0 \leq \gamma < 1$ is the discount factor. Since observations generally do not reveal the full state of the environment, the classical methods often maintain a belief over possible states, $\text{bel}(s_t) \triangleq p(s_t|o_{\leq t}, a_{<t})$, and update the belief according to

$$\text{bel}(s_{t+1}) = \eta \int \text{bel}(s_t)\mathcal{Z}(o_{t+1}|s_{t+1})\mathcal{T}(s_{t+1}|s_t, a_t)ds_t, \quad (1)$$

where η is a normalization factor. In this work, we make the true states available during training only as a supervised signal for the filter and keep them unobserved during testing. The key to solve continuous POMDPs is to perceive the state uncertainty and make decisions under such uncertainty.

3 Related Work

Planning under uncertainty. Due to the high computation cost of POMDPs, many previous approaches used sampling-based techniques for either belief update or planning, or both. For instance, a variety of *Monte Carlo tree search* methods have shown success in relatively large POMDPs by constructing a search tree of history based on rollout simulations [Silver and Veness, 2010; Somani *et al.*, 2013; Seiler *et al.*, 2015; Sunberg and Kochenderfer, 2018]. Later work further improved the efficiency by limiting the search space or reusing plans [Somani *et al.*, 2013; Kurniawati and Yadav, 2016]. Although considerable progress has been made to enlarge the set of solvable POMDPs, it remains hard for pure sampling-based methods to deal with unknown dynamics and complex observations like visual inputs. Therefore, in this work, we provide one approach to combine the efficiency and interpretability of conventional sampling-based methods with the flexibility of deep learning networks for complex POMDP modeling.

Differentiable particle filter. Ever since its invention [Gordon *et al.*, 1993], the Particle Filter (PF), or Sequential Monte Carlo (SMC), has become a well-suited method for sequential estimation in complex non-linear scenarios. A large number of research has made progress on learning a flexible proposal distribution for SMC[†]. Gu *et al.* [2015] was one of the earliest that use a recurrent neural network to model the proposal distribution. Naesseth *et al.* [2018] and Maddison *et al.* [2017] further provided a variational framework that learns a good parameterized proposal distribution by optimizing the log estimator. Recently, Karkus *et al.* [2018] and Jonschkowski *et al.* [2018] introduced differentiable particle filters independently and applied them to localization problems with rich visual input. These approaches explicitly treat the proposal distribution as three interleaved neural modules: a proposer that generates plausible states, a transition model that simulates dynamics, and an observation model that does Bayesian belief update. The filter in our model is based on [Jonschkowski *et al.*, 2018], with an additional adversarial objective. Kempinska and Shawe-Taylor [2017] also proposed an adversarial training objective for SMC. But their objective is for learning the proposal distribution, while our method focuses more on mutually enhancing the proposer and observation model.

Planning as inference. The framework of *control as probabilistic inference* considers that selecting the optimal action is equivalent to finding the maximum posterior over actions conditioned on an optimal future [Todorov, 2008; Toussaint, 2009; Kappen *et al.*, 2012; Levine and Koltun, 2013]. We refer to [Levine, 2018] as an explanatory review of these methods. Piche *et al.* [2018] extended this idea further to planning problems and propose the sequential Monte Carlo planning (SMCP), where the inference is done over optimal future trajectories. While most previous work focused on Markov Decision Processes (MDP) with full observation, we take one step further and apply the *planning as inference* framework to POMDP problems. On the other hand, compared with the existing Bayesian reinforcement learning literature on POMDPs [Ross *et al.*, 2008], our work focuses more on deep reinforcement learning solutions to continuous POMDPs.

4 Dual Sequential Monte Carlo Network

In this section, we first introduce the adversarial particle filter that aims to mutually enhance the particle proposer model and the observation model. Then, we illustrate the design choice of our main algorithm and describe in detail how our method connects the two SMCs, during which we also introduce an alternative simpler formulation for SMCP [Piche *et al.*, 2018].

4.1 Adversarial Particle Filtering

A particle filter represents a *belief distribution* $\text{bel}(s_t)$ of the true state s_t with a set of weighted particles $\{(s_t^{(k)}, w_t^{(k)})\}_{k=1}^K$, where $\sum_{k=1}^K w_t^{(k)} = 1$. To perform Bayesian update when action is applied and a new observation comes in, it first transits

[†]The proposal distribution refers to the posterior distribution over the latent variables in an SMC. This should not be confused with the particle proposer model in this paper, which is a separate model that proposes possible state particles given observations.

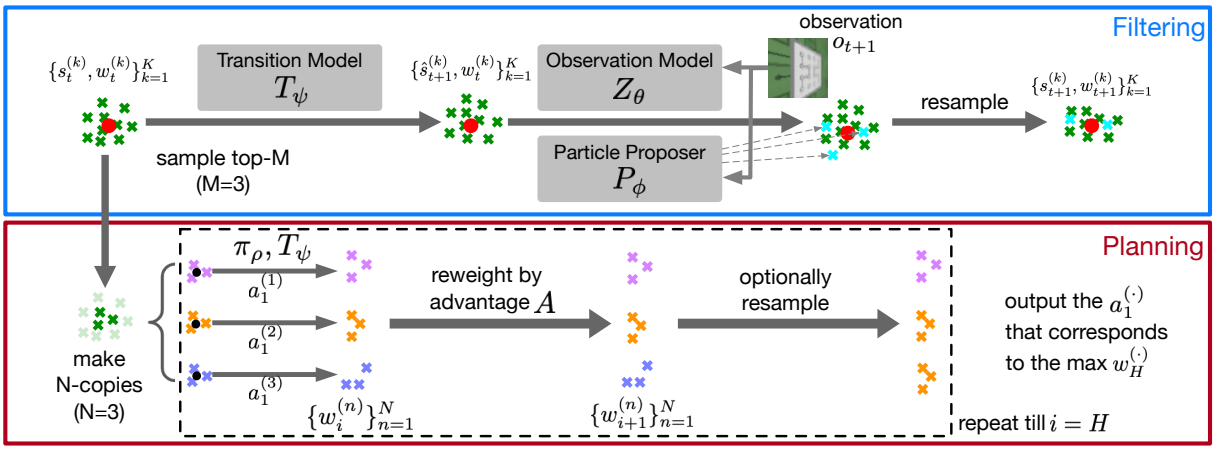


Figure 1: A schematic drawing of the modules in DualSMC. Here we choose $M = 3$ and $N = 3$ for illustration

all particles according to a transition model and then update corresponding weights according to an observation model:

$$s_{t+1}^{(k)} \sim \mathcal{T}(\cdot|s_t^{(k)}, a_t) \text{ and } w_{t+1}^{(k)} \propto \mathcal{Z}(o_{t+1}|s_{t+1}^{(k)}) w_t^{(k)}. \quad (2)$$

In practice, when the true dynamics $\mathcal{T}$ and $\mathcal{Z}$ are not known a priori, they can be approximated by the parameterized functions $T_\psi(\cdot)$ and $Z_\theta(\cdot)$. Similar to [Jonschkowski *et al.*, 2018], our differentiable particle filter contains three neural modules (Figure 1): the proposer $P_\phi(o_t, \epsilon_P)$, the transition model $T_\psi(s_{t-1}^{(k)}, a_{t-1}, \epsilon_T)$, and the observation model $Z_\theta(o_t, s_t^{(k)})$, where ϕ, ψ, θ are parameters. ϵ_P and ϵ_T are the Gaussian noises for stochastic models. To avoid the particle degeneracy problem [Doucet and Johansen, 2009], we perform Sequential Importance Resampling (SIR) together with the proposer model. Specifically, after the Bayesian update at time t, we sample K' old particles $\{s_{\text{old}}^{(k)}\}_{k=1}^{K'}$ with replacement based on the updated weight and combine them with $(K - K')$ newly proposed particles $\{s_{\text{new}}^{(k)}\}_{k=K'+1}^{K}$, and assign uniform weights for all particles. Depending on the task, we keep K' constant or make $(K - K')$ follow an exponential decay.

The major difference between our filtering approach and [Jonschkowski *et al.*, 2018] comes in by noticing that P_ϕ and Z_θ are naturally opposite to yet dependent on each other. Following this intuition, instead of regressing the output of the proposer to the true state, we propose the adversarial proposing strategy. In particular, we train Z_θ to differentiate the true state from all particle states and train P_ϕ to fool Z_θ. Formally, denote $p_{\text{real}}(o_{\le t}), p_{\text{real}}(s|o_{\le t})$ as the real distributions over observations and the real posterior over s, Z_θ and P_ϕ play the following two-player minimax game with function $F(Z_\theta, P_\phi)$:

$$\begin{aligned} \min_\phi \max_\theta F(Z_\theta, P_\phi) = \mathbb{E}_{o_{1:t} \sim p_{\text{real}}(o_{\le t})} \Big[\\ \mathbb{E}_{s \sim p_{\text{real}}(s|o_{\le t})} \log Z_\theta(o_t, s) + \\ \mathbb{E}_{s' \sim s_{\text{old}}^{(k)}} \log(1 - Z_\theta(o_t, s')) + \\ \mathbb{E}_{\epsilon_P \sim \mathcal{N}(0,I)} \log(1 - Z_\theta(o_t, P_\phi(o_t, \epsilon_P))) \Big]. \end{aligned} \quad (3)$$

During training, instead of using trajectories sampled from a random or heuristic policy [Jonschkowski *et al.*, 2018; Karkus *et al.*, 2018], we train the filter in an on-policy way so that it can take advantage of the gradually more powerful planner.

Algorithm 1 Overall DualSMC algorithm

1: $\{s_1^{(k)} \sim \text{Priori}(s_1)\}_{k=1}^K, \{w_0^{(k)} = 1\}_{k=1}^K$
2: **for** $t = 1 : L$ **do**
3: // At each filtering and control step
4: $\{w_t^{(k)} \propto w_{t-1}^{(k)} \cdot Z_\theta(s_t^{(k)}, o_t)\}_{k=1}^K$
5: $\overline{\text{bel}}_t = \sum_k w_t^{(k)} s_t^{(k)}$
6: $\{\tilde{s}_t^{(m)}, \tilde{w}_t^{(m)}\}_{m=1}^M = \text{Top-M}(\{s_t^{(k)}, w_t^{(k)}\}_{k=1}^K)$, w.r.t.$\{w_t^{(k)}\}_k$
7: $a_t = \textbf{DualSMC-P}(\overline{\text{bel}}_t, \{\tilde{s}_t^{(m)}, \tilde{w}_t^{(m)}\}_{m=1}^M; \pi_\rho, Q_\omega)$
8: $o_{t+1}, r_t \sim p_{\text{env}}(a_t)$
9: **if** resample **then**
10: $\{s_t^{(k)}\}_{k=1}^{K'} \sim \text{Multinomial}(\{s_t^{(k)}\}_{k=1}^K)$, w.r.t.$\{w_t^{(k)}\}_k$
11: $\{s_t^{(k)} \sim P_\phi(o_t)\}_{k=K'+1}^K, \{w_t^{(k)} = 1\}_{k=1}^K$
12: **end if**
13: $\{s_{t+1}^{(k)} \sim T_\psi(s_t^{(k)}, a_t)\}_{k=1}^K$
14: Add $(s_t, s_{t+1}, a_t, r_t, o_t, \overline{\text{bel}}_t, \{\tilde{s}_t^{(m)}, \tilde{w}_t^{(m)}\}_{m=1}^M)$ to a buffer
15: Sample a batch from the buffer and update $(\rho, \omega, \theta, \psi, \phi)$
16: **end for**

4.2 DualSMC Planning on Explicit Belief States

A straightforward solution to POMDP planning is to train the planning module separately from the filtering module. At inference time, plans are made *independently based on each particle state*. We thus name this planning algorithm the Particle-Independent SMC Planning (PI-SMCP) and use it as a baseline method. More details on PI-SMCP can be found in Appendix A. Although PI-SMCP is unbiased, it does not perform well in practice because it cannot generate policies based on dynamically varying state uncertainties.

We thus propose the DualSMC algorithm to explicitly consider the belief distribution by planning directly on an approximated belief representation, i.e., a combination of the top candidates from the filter (for computation efficiency) as well as the weighted mean estimate. We show the modules in DualSMC and how they relate to each other in Figure 1.

The overall algorithmic framework of DualSMC is shown in Alg 1. At time step t, when a new observation comes, we first use the observation model Z_θ to update the particle weights

Algorithm 2 DualSMC planner on estimated belief states

Input: $\overline{\text{bel}}_t$, $\{\tilde{s}_t^{(m)}, \tilde{w}_t^{(m)}\}_{m=1}^M$
Output: a_t

1: $\{\tilde{w}_t^{(m)}\}_{m=1}^M = \text{Normalize}(\{\tilde{w}_t^{(m)}\}_{m=1}^M)$
2: $\big\{\{\hat{s}_t^{(m)(n)} = \tilde{s}_t^{(m)}\}_{m=1}^M, \hat{w}_{t-1}^{(n)} = 1, \overline{\text{bel}}_t^{(n)} = \overline{\text{bel}}_t\big\}_{n=1}^N$
3: **for** $i = t : t+H$ **do**
4: // At each planning time step
5: $\big\{a_i^{(n)} \sim \pi_\rho(\{\hat{s}_i^{(m)(n)}\}_{m=1}^M, \overline{\text{bel}}_i^{(n)})\big\}_{n=1}^N$
6: $\big\{\hat{s}_{i+1}^{(m)(n)}, r_i^{(m)(n)} \sim T_\psi(\hat{s}_i^{(m)(n)}, a_i^{(n)})\big\}_{m=1,n=1}^{M,N}$
7: $\big\{\overline{\text{bel}}_{i+1}^{(n)} = \sum_m \tilde{w}_t^{(m)} \hat{s}_{i+1}^{(m)(n)}\big\}_{n=1}^N$
8: $\big\{\hat{w}_i^{(n)} \propto \hat{w}_{i-1}^{(n)} \cdot \exp\big(\sum_m \tilde{w}_t^{(m)} A^{(m)(n)}\big)\big\}_{n=1}^N$
9: $\big\{x_i^{(n)} = (\{\hat{s}_{i+1}^{(m)(n)}, \hat{s}_i^{(m)(n)}\}_{m=1}^M, \overline{\text{bel}}_{i+1}^{(n)}, a_i^{(n)})\big\}_{n=1}^N$
10: **if** resample **then**
11: $\{x_{t:i}^{(n)}\}_{n=1}^N \sim \text{Multinomial}(\{x_{t:i}^{(n)}\}_{n=1}^N)$, w.r.t. $\{\hat{w}_i^{(n)}\}_n$
12: $\big\{\hat{w}_i^{(n)} = 1\big\}_{n=1}^N$
13: **end if**
14: **end for**
15: a_t = first action of $x_{t:t+H}^{(n)}$, where $n \sim \text{Uniform}(1, \ldots, N)$

(line 4 in Alg 1), and then perform the DualSMC planning algorithm in Alg 2. We duplicate the top-M particles (for computation efficiency) and the mean belief state N times as the root states of N planning trajectories (line 1-2 in Alg 2). Different from the previous SMCP [Piche *et al.*, 2018] method under full observations, the policy network π_ρ perceives the belief states and predicts an action based on the top-M particle states as well as the mean belief state (line 5 in Alg 2). We then perform N actions to $M \times N$ states and use T_ψ to predict the next states and rewards (line 6 in Alg 2). Since future observations $o_{>t}$ are not available at current time step, inspired by QMDP [Littman *et al.*, 1995], we assume the uncertainty disappears at the next step, and thus approximate $\overline{\text{bel}}_{i>t}^{(n)}$ using the top-M transition states as well as a set of fixed filtering weights (line 7 in Alg 2). We update the planning weight of each planning trajectory by summarizing the advantages of each state using the initial M belief weights (line 8 in Alg 2). Here, we introduce an alternative advantage formulation that is equivalent to the one used in [Piche *et al.*, 2018]:

$$\text{TD}_{i-1}^{(m)(n)} = Q_\omega(\hat{s}_i^{(m)(n)}, a_i^{(n)}) - Q_\omega(\hat{s}_{i-1}^{(m)(n)}, a_{i-1}^{(n)}) + r_{i-1}^{(m)(n)},$$
$$A^{(m)(n)} = \text{TD}_{i-1}^{(m)(n)} - \log \pi_\rho(a_i^{(n)} | \{\hat{s}_i^{(m)(n)}\}_{m=1}^M, \overline{\text{bel}}_i^{(n)}). \tag{4}$$

At time t, $Q_\omega(\hat{s}_{i-1}^{(m)(n)}, a_{i-1}^{(n)})$ and $r_{i-1}^{(m)(n)}$ are set to 0. It is a benefit, because according to Eq. (3.3) in the SMCP paper [Piche *et al.*, 2018], the advantage in SMCP depends on both Q and the log expectation of V, which can be difficult to estimate accurately, while our approach only requires Q, which is much simpler. We leave the full derivation to Appendix B.

At the end of each planning time step, we may apply resampling when necessary over N planning trajectories[‡] (line 11-12 in Alg 2.). When the planning horizon is reached, where $i = t + H$, we sample one planning trajectory (line 15 in Alg 2) and feed its first action to the environment. We then go back to the filtering part and update the belief states by resampling, proposing, and predicting the next-step particle states (line 9-13 in Alg 1). Lastly, we train all modules of DualSMC, including the policy network, the critic network, and the three modules of the adversarial particle filter (line 14-15 in Alg 1).

[‡]The resampling steps in both filtering and planning algorithms may not be performed at every time step (more details in Table 1).

Hyper-parameter	A	B	C
Training episodes	10,000	2,000	5,000
Learning rate	0.001	0.0003	0.0003
Batch size	64	128	256
Filtering particles	100	100	100
- Resampling frequency	8	3	2
Planning trajectories	30	10	10
- Planning time horizon	10	10	1
- Resampling frequency	3	1	1

Table 1: Training hyper-parameters for the (A) floor positioning, (B) 3D dark-light, and (C) modified reacher domains

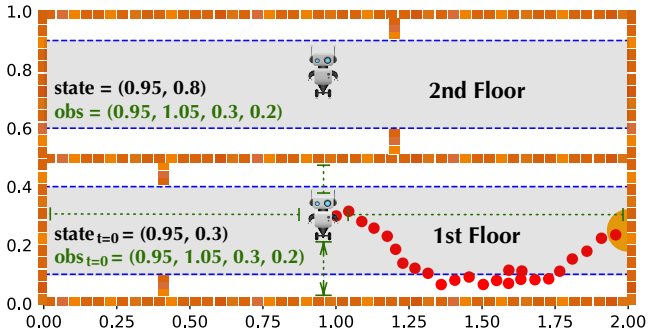


Figure 2: An illustration of the floor positioning domain

5 Experiment

Our experiments are designed in the following way. First, we use a 2D Floor Positioning task to illustrate the effectiveness of both the adversarial proposing and that of the uncertainty-dependent planning. Next, we test the DualSMC network on a much harder navigation task with both high uncertainty and visual input. At least, we present a modified Reacher environment to further test DualSMC's performance beyond the navigation domain. All models are trained with the Adam optimizer [Kingma and Ba, 2015]. The training hyper-parameters are shown in Table 1. We tuned the number of filtering particles as it largely determines the quality of belief state estimation, which is the foundation of the subsequent DualSMC planning algorithm. After many trials, we finally set it to 100 for a balance between planning results and efficiency. All experimental results are averaged over 5 runs of training. Network details can be found in Appendix C.

5.1 Floor Positioning

Suppose there is a robot in a two-floor building as shown in Figure 2, who doesn't know which floor it resides on. It can only distinguish different floors by observation.

- State: It is defined as the robot's position in world coordinates (s_x, s_y), i.e., the axes in Figure 2.

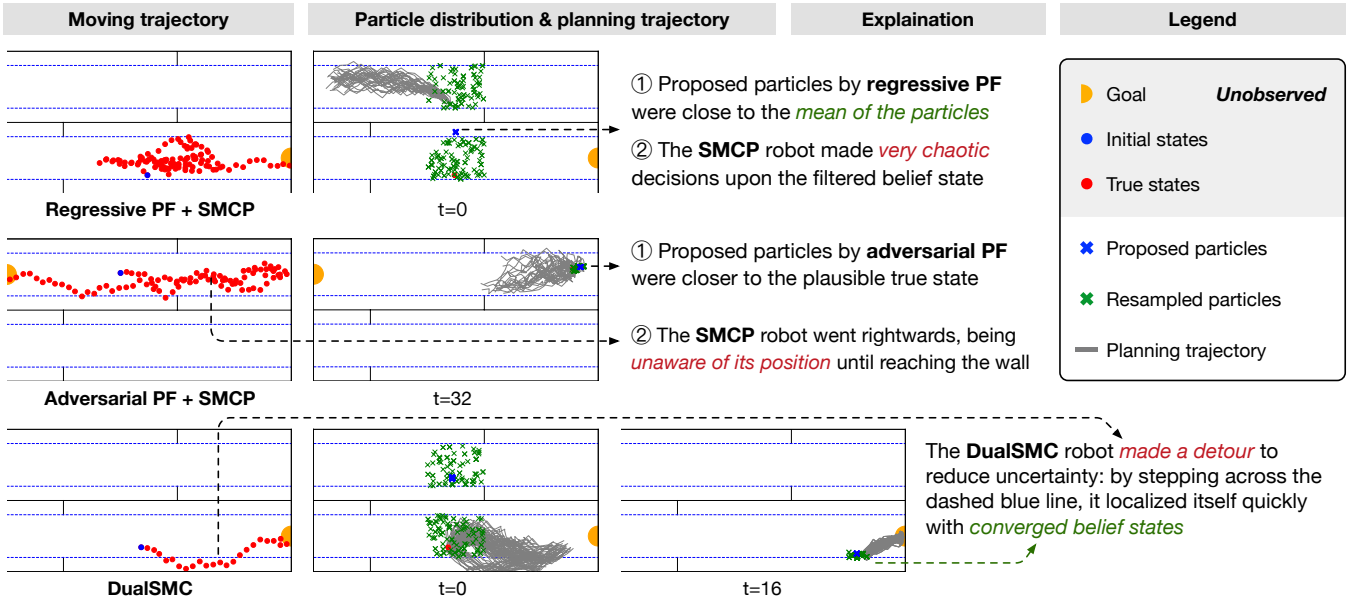


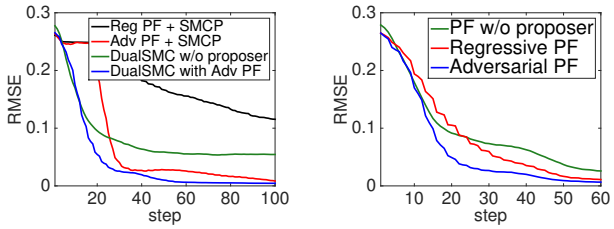
Figure 3: Qualitative results in the floor positioning domain, including the robot's actual moving trajectories and its planning trajectories

Method	Success	# Steps
DVRL [Igl *et al.*, 2018]	38.3%	162.0
LSTM filter + SMCP [Piche *et al.*, 2018]	23.5%	149.1
Regressive PF (ℓ_2, top-1) + SMCP	25.0%	107.9
Regressive PF (density, top-3) + PI-SMCP	25.0%	107.9
Adversarial PF (top-1) + SMCP	95.0%	73.3
Adversarial PF (top-3) + PI-SMCP	82.7%	86.9
DualSMC with regressive PF (ℓ_2)	45.1%	114.9
DualSMC with regressive PF (density)	58.3%	107.0
DualSMC w/o proposer	78.6%	62.1
DualSMC with adversarial PF	**99.4%**	**26.9**

Table 2: The success rate and the average number of steps of 1,000 tests in the floor positioning domain (PF is short for particle filter)

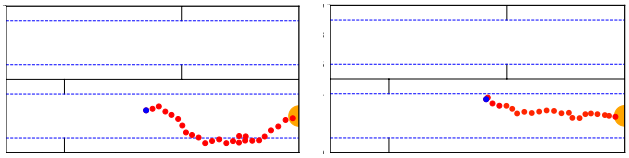
- Action: It is defined as $a = (\Delta s_x, \Delta s_y)$ with a maximum magnitude of 0.05.
- Observation: It is defined as the robot's horizontal distances to the nearest left/right walls, and the vertical distances to ceiling/ground $o_t = (d_{x-}, d_{x+}, d_{y-}, d_{y+})_t$. In the case of Figure 2, it starts with an observation of $(0.95, 1.05, 0.3, 0.2)$, whatever floor it is on.
- Goal: The robot starts from a random position and is headed to different regions according to different floors. If it is on the first floor, the target area is around $(2, 0.25)$ orange semicircle area); If the robot is on the second floor, the target area is around $(0, 0.75)$. Only at training time, a reward of 100 is given at the end of each episode if the robot reaches the correct target area.

Starting from a gray area, the robot is very uncertain about its y-axis position. In the case of Figure 2, the estimated state can be $(0.95, 0.3)$ or $(0.95, 0.8)$. Only when the robot goes across a dashed blue line, from the gray area to the bright one, does it become certain about its y-axis position.



(a) Different POMDP planners (b) Different particle filters

Figure 4: The state filtering error with respect to the number of steps which the robot has taken in the floor positioning domain



(a) Partial observation (b) Full observation

Figure 5: The DualSMC planner generates different polices based on the uncertainty of the perceived belief state

Learned policies. The first two rows in Figure 3 show the planning results by applying the SMCP algorithm [Piche *et al.*, 2018] to the top-1 estimated particle state. Training the proposer with the mean squared loss is equivalent to regressing the proposed particles to the mean values of the multi-modal state distributions under partial observations. Thus, in the first example, the robot cannot make reasonable decisions due to incorrect estimation of plausible states. The second row in Figure 3 shows the moving and planning trajectories by using an adversarial *particle filter* (PF), which leads the proposed particle states closer to plausible states. The robot learns an interesting policy (along a path marked by red dots): *it always goes rightwards at first, being unaware of its position until it reaches the wall, and then bouncing back at the wall*. However, this policy is suboptimal, as it does not fully consider

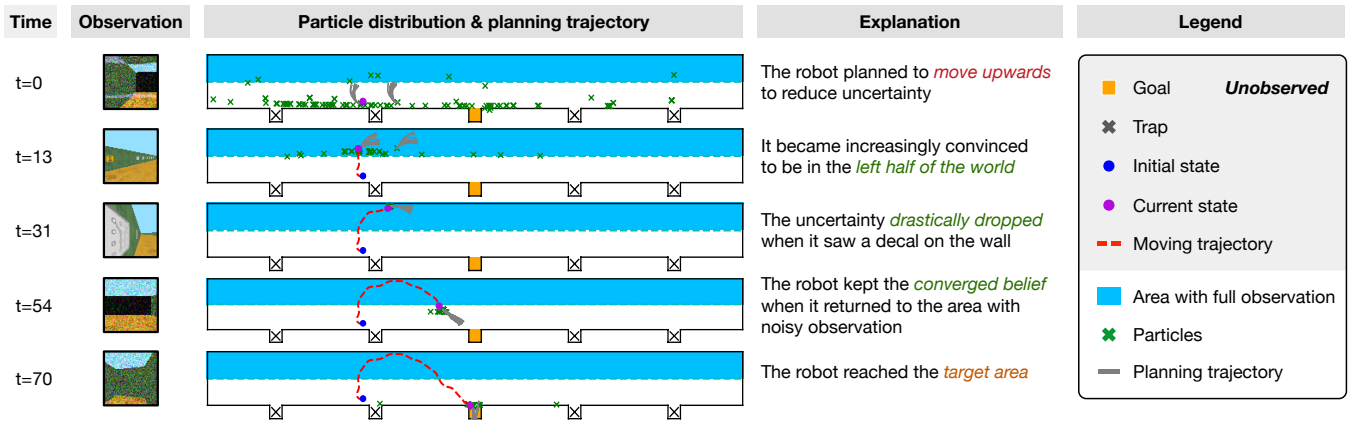


Figure 6: A demonstration trajectory from DualSMC with an *adversarial filter* on the 3D light-dark navigation task

the uncertainty of the belief state. In contrast, DualSMC has learned to *reduce uncertainty by making short detours at first*, as shown by the last row in Figure 3. We have three findings. First, the robot learns to localize itself quickly and then approach the target area in fewer steps. Second, the adversarial PF works well: once the robot steps across the dashed blue line, the belief states quickly converge to the actual values, and the observation model maintains its confidence in the converged belief even when the robot moves back to the middle areas. Third, DualSMC generates probabilistic planning trajectories of moving up/down with different advantage values.

Quantitative comparisons. From Table 2, the final DualSMC model takes 26.9 steps to reach the target area, whilst the baseline model "Adversarial PF + SMCP" uses as many as 73.3 steps on average. Besides, we can see that the adversarial PF significantly outperforms other differentiable state estimation approaches, such as (1) the existing DPFs that perform density estimation [Jonschkowski *et al.*, 2018], and (2) the deterministic LSTM model that was previously used as a strong baseline in [Karkus *et al.*, 2018; Jonschkowski *et al.*, 2018]. Also note that DualSMC models with regressive proposers are even worse than one without any proposer, which suggests that an inappropriate proposer may cause a negative effect on solving continuous POMDPs.

Does the adversarial training improve the DPF? Given partial observations, an ideal filter should derive a complete distribution of possible states instead of point estimation. Figure 4(a) compares the average RMSE between the true states and the filtered states by different models. The adversarial PF performs best, while the PF with the regressive proposer performs even worse than that without a proposer. A natural question arises: as the filtering error is also related to different moving trajectories of different models, can we eliminate this interference? For Figure 4(b), we train different filters without a planner. All filters follow the same expert trajectories, and the adversarial PF still achieves the best performance.

How does DualSMC adapt to different uncertainties? In a fully observable scenario, we suppress the filtering part of DualSMC and assume DualSMC plans upon a converged belief on the true state (s_x, s_y). That is to say, we take the *true state* to as the top-M particles (line 7 in Alg 1) before the planning part. The robot changes its plan from taking a detour shown in Figure 5(a) to walking toward the target area directly shown in Figure 5(b). It performs equally well to the standard SMCP, with a 100.0% success rate and an averaged 21.3 steps (v.s. 20.7 steps by SMCP). We may conclude that DualSMC provides policies based on the distribution of filtered particles. We may also conclude that DualSMC trained under POMDPs generalizes well to similar tasks with less uncertainty.

Method	Success	# Steps
PlaNet [Hafner *et al.*, 2019]	30%	34.24
DVRL [Igl *et al.*, 2018]	42%	98.48
LSTM + SMCP [Piche *et al.*, 2018]	59%	85.40
Adversarial PF (top-1) + SMCP	58%	56.11
Adversarial PF (top-3) + PI-SMCP	64%	64.37
DualSMC with regressive PF (ℓ_2)	92%	66.88
DualSMC with regressive PF (density)	98%	70.95
DualSMC with adversarial PF	**98%**	**67.49**

Table 3: The average result of 100 tests for 3D light-dark navigation

5.2 3D Light-Dark Navigation

We extend the 2D light-dark navigation domain [Platt Jr *et al.*, 2010] to a visually rich environment simulated by DeepMind Lab [Beattie *et al.*, 2016]. At the beginning of each episode, the robot is placed randomly and uniformly on one of the four platforms at the bottom (see Figure 6). The robot's goal is to navigate toward the central cave (marked in orange) while avoiding any of the four traps (marked by crosses). The maze is divided into upper and lower parts. Within the lower part, the robot travels in darkness, receives noisy visual input of a limited range (up to a fixed depth), and therefore suffers from high state uncertainty. When the robot gets to the upper part (the blue area), it has a clear view of the entire maze. We place decals as visual hints on the top walls of the maze to help the robot figure out its position. However, it has to be very close to the upper walls to see clearly what these decals are. The robot receives a positive reward of 100 when it reaches the goal and a negative reward of -100 when in a trap. At each time step, the robot's observation includes a 64×64 RGB image, its current velocity, and its orientation. We force it to move forward and only control its continuous orientation.

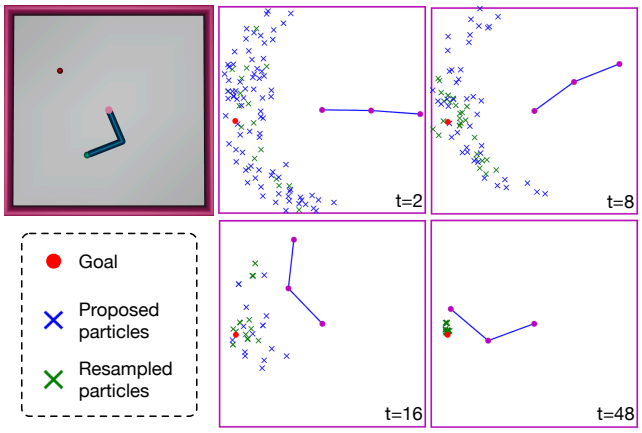


Figure 7: The modified Reacher environment and examples of the posterior belief over states given by an adversarial particle filter

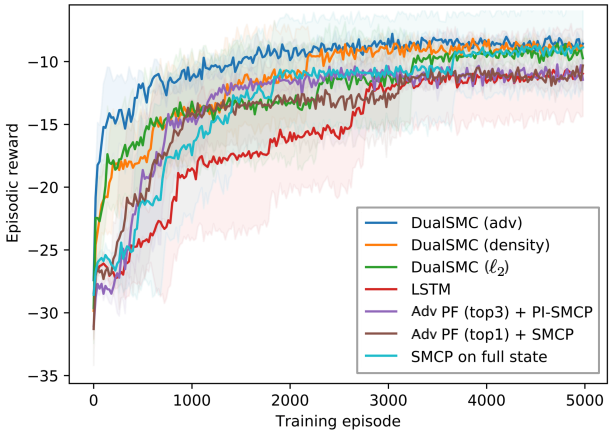


Figure 8: Training curves of DualSMC and baseline methods for the modified Reacher environment (averaged over 5 seeds)

By considering the uncertainty, DualSMC methods outperform other baselines in success rate (see Table 3). An excessively large number of steps indicates that the robot is easy to get lost while too few steps means that it is easy to fall into a trap. From Figure 6, DualSMC is the only one that learned to go up and figure out its position first before going directly towards the goal.

5.3 Modified Reacher

We further validate our model on a continuous control task with partial observation, i.e., a modified Reacher environment from OpenAI Gym [Brockman *et al.*, 2016]. The original observation of Reacher is a 11-D vector including $(\cos\theta_1, \cos\theta_2, \sin\theta_1 \sin\theta_2, g_x, g_y, \omega_1, \omega_2, r_x, r_y, r_z)$, where the first 4 dimensions are cos/sin values of the two joint angles θ_1, θ_2, g_x, g_y the goal position, ω_1, ω_2 the angular velocities and r_x, r_y, r_z the relative distance from the end-effector to the goal. We remove g_x, g_y, r_x, r_y, r_z from the original observation and include a single scalar $r = ||(r_x, r_y, r_z)||_2 + \epsilon_r$, where $\epsilon_r \sim \mathcal{N}(0, 0.01)$ is a small noise (r is usually on the scale of 0.1). The observation is therefore a 7-D vector. The robot has to simultaneously locate the goal and reach it.

We provide a visualization of one sample run under DualSMC with the adversarial filter in Figure 7. As expected, initially the proposed particles roughly are in a half-cycle and as time goes on, the particles gradually concentrate around the true goal. Since the final performance of various methods is similar after long enough time of training, we provide the training curve of these methods in Figure 8, and truncate the results up to 5,000 episodes since no obvious change in performance is observed from thereon. As we can see, the DualSMC methods not only achieve similar asymptotic performance as the SMCP method with full observation but also learn faster to solve the task than baseline methods.

6 Conclusion

In this paper, we provided an end-to-end neural network named DualSMC to solve continuous POMDPs, which has three advantages. First, it learns plausible belief states for high-dimensional POMDPs with an adversarial particle filter. For simplicity, we use the naïve adversarial training method from the original GANs [Goodfellow *et al.*, 2014]. One may potentially improve DualSMC with modern techniques to stabilize training and lessen mode collapse. Second, DualSMC plans future actions by considering the distributions of the learned belief states. The filter module and the planning module are jointly trained and facilitate each other. Third, DualSMC combines the richness of neural networks as well as the interpretability of classical sequential Monte Carlo methods. We empirically validated the effectiveness of DualSMC on different tasks including visual navigation and control.

Acknowledgments

This work is in part supported by ONR MURI N00014-16-1-2007.

A Particle-Independent SMC Planning

As shown in Alg 3, it takes the top-M particle states (for computation efficiency) and plans N future trajectories *independently based on each particle state*. At the end of the planning horizon H, it samples a trajectory from $M \times N$ planning trajectories. Although PI-SMCP is unbiased, it does not perform well in practice because it cannot generate policies based on dynamically varying state uncertainties.

B A Simpler Formulation of SMC Planning

At time t, we set $Q_\omega(\hat{s}_{i-1}^{(m)(n)}, a_{i-1}^{(n)})$ and $r_{i-1}^{(m)(n)}$ in Eq. (4) to 0. We emphasize that our formulation is much simpler than the original SMCP [Piche *et al.*, 2018]. Because it only requires a learned Q function and more importantly, it prevents us from estimating the expectation of the value function V. To prove this, we depict the Hidden Markov Model of our planning algorithm for ease of notation. Figure 9 is borrowed from [Piche *et al.*, 2018]. $\mathcal{O}_t$ is a convenience binary variable here for the sake of modeling, denoting the "optimality" (optimal policy) of a pair (s_t, a_t) at time t [Levine, 2018]. Then we present the derivation of line 8 Alg 2 as follows. Comparing with [Piche *et al.*, 2018], our update of the planning particle weights depends only on the learned Q and π_ρ.

Module	Layers (floor positioning)	# Channels	Layers (3D dark-light)	# Channels	Filter	Stride
Z_θ	3-layer MLP	$256 \times 2, 16$	Conv2d×2, MaxPool(2)	$16, 32$	$(3, 3)$	2
			Conv2d, Dropout(0.2) (*)	64	$(3, 3)$	2
			Fully connected	64		
	2-layer LSTM	128×2	2-layer LSTM	64×2		
			Concat: state, orientation	67		
	3-layer MLP	$256 \times 2, 1$	4-layer MLP	$128 \times 3, 1$		
P_ϕ	3-layer MLP	$256 \times 2, 64$	Same as Z_θ up to (*)	64	$(3, 3)$	2
			Fully connected	64		
	Concat: $z(64) \sim \mathcal{N}(0, 1)$	128	Concat: $z(64) \sim \mathcal{N}(0, 1)$, orientation	129		
	4-layer MLP	$256 \times 3, 2$	3-layer MLP	$128 \times 3, 2$		
T_ψ	4-layer MLP	$256 \times 3, 4$	Action noise $\sim \mathcal{N}(0, 1)$	1		
			3-layer MLP: encode action noise to e	$128 \times 2, 1$		
			Concat: (state, action + e)	6		
			3-layer MLP, then add to state	$128 \times 3, 5$		
Q_ω	3-layer MLP	$256 \times 2, 1$	3-layer MLP	$128 \times 2, 1$		
π_ρ	3-layer MLP	$256 \times 2, 4$	3-layer MLP	$128 \times 2, 1$		

Table 4: Network details of each module in DualSMC

C Network Details

Table 4 shows the network details for the floor positioning domain and the 3D dark-light domain.

Algorithm 3 Particle-Independent SMC Planning

Input: $\{\tilde{s}_t^{(m)}, \tilde{w}_t^{(m)}\}_{m=1}^M$
Output: a_t

$\{\hat{s}_t^{(m\times n)} = \tilde{s}_t^{(m)}, \hat{w}_t^{(m\times n)} = \tilde{w}_t^{(m)}\}_{m=1,n=1}^{M,N}$
$\{\hat{w}_t^{(n)}\}_{n=1}^{MN} = \text{Normalize}(\{\hat{w}_t^{(n)}\}_{n=1}^{MN})$
for $i = t : t + H$ **do**
  // Predict actions based on individual particle states
  $\{a_i^{(n)} \sim \pi_\rho(\hat{s}_i^{(n)})\}_{n=1}^{MN}$
  $\{\hat{s}_{i+1}^{(n)}, r_i^{(n)} \sim T_\psi(\hat{s}_i^{(n)}, a_i^{(n)})\}_{n=1}^{MN}$
  $\{\hat{w}_{i+1}^{(n)} \propto \hat{w}_i^{(n)} \cdot \exp(A(\hat{s}_i^{(n)}, a_i^{(n)}, \hat{s}_{i+1}^{(n)}))\}_{n=1}^{MN}$
  $\{x_i^{(n)} = (\hat{s}_{i+1}^{(n)}, a_i^{(n)}, \hat{s}_i^{(n)})\}_{n=1}^{MN}$
  if resample **then**
    $\{x_{t:i}^{(n)}\}_{n=1}^{MN} \sim \text{Multinomial}(\{x_{t:i}^{(n)}\}_{n=1}^{MN}), \text{w.r.t.} \{\hat{w}_{i+1}^{(n)}\}_n$
    $\{\hat{w}_{i+1}^{(n)} = 1\}_{n=1}^{MN}$
  end if
end for
$a_t =$ first action of $x_{t:t+H}^{(n)}, n \sim \text{Uniform}(1, \ldots, MN)$

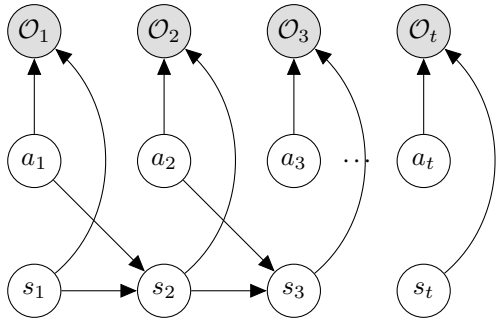


Figure 9: $\mathcal{O}_t$ is the observed *optimality* variable with probability $p(\mathcal{O}_t|s_t, a_t) = \exp(r(s_t, a_t))$, where $r(s, a)$ is the reward function

$$
\begin{aligned}
w_t &= \frac{p(x_{1:t}|\mathcal{O}_{1:T})}{q(x_{1:t})} \\
&= \frac{p(x_{1:t-1}|\mathcal{O}_{1:T})}{q(x_{1:t-1})} \frac{p(x_t|x_{1:t-1}, \mathcal{O}_{1:T})}{q(x_t|x_{1:t-1})} \\
&= w_{t-1} \frac{p(x_t|x_{1:t-1}, \mathcal{O}_{1:T})}{q(x_t|x_{1:t-1})} \\
&= \frac{w_{t-1}}{q(x_t|x_{1:t-1})} \frac{p(x_{1:t}|\mathcal{O}_{1:T})}{p(x_{1:t-1}|\mathcal{O}_{1:T})} \\
&= \frac{w_{t-1}}{q(x_t|x_{1:t-1})} \frac{p(\mathcal{O}_{1:T}|x_{1:t})p(x_{1:t})}{p(\mathcal{O}_{1:T}|x_{1:t-1})p(x_{1:t-1})} \\
&= \frac{w_{t-1}}{q(x_t|x_{1:t-1})} \frac{p(\mathcal{O}_{1:t-1}|x_{1:t-1})p(x_{1:t})p(\mathcal{O}_{t:T}|x_t)}{p(\mathcal{O}_{1:t-2}|x_{1:t-2})p(x_{1:t-1})p(\mathcal{O}_{t-1:T}|x_{t-1})} \\
&= \frac{w_{t-1}}{q(x_t|x_{1:t-1})} p(x_t|x_{t-1})p(\mathcal{O}_{t-1}|x_{t-1}) \\
&\quad \exp\big(Q(s_t, a_t) - Q(s_{t-1}, a_{t-1})\big) \\
&= w_{t-1} \frac{p(x_t|x_{t-1})}{q(x_t|x_{t-1})} \exp\big(Q(s_t, a_t) - Q(s_{t-1}, a_{t-1}) + r_{t-1}\big) \\
&= w_{t-1} \frac{p_{env}(s_t|s_{t-1}, a_{t-1})}{p_{model}(s_t|s_{t-1}, a_{t-1})} \\
&\quad \exp\big(Q(s_t, a_t) - Q(s_{t-1}, a_{t-1}) + r_{t-1} - \log \pi_\rho(a_t|s_t)\big).
\end{aligned} \tag{5}
$$

References

[Beattie *et al.*, 2016] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. DeepMind Lab. *arXiv preprint arXiv:1612.03801*, 2016.

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[Doucet and Johansen, 2009] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fif-

teen years later. *Handbook of Nonlinear Filtering*, 12(656-704):3, 2009.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, pages 2672–2680, 2014.

[Gordon *et al.*, 1993] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, pages 107–113, 1993.

[Gu *et al.*, 2015] Shixiang Shane Gu, Zoubin Ghahramani, and Richard E Turner. Neural adaptive sequential Monte Carlo. In *NeurIPS*, pages 2629–2637, 2015.

[Hafner *et al.*, 2019] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *ICML*, pages 2555–2565, 2019.

[Hausknecht and Stone, 2015] Matthew Hausknecht and Peter Stone. Deep recurrent Q-learning for partially observable MDPs. In *2015 AAAI Fall Symposium Series*, 2015.

[Igl *et al.*, 2018] Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for POMDPs. In *ICML*, pages 2117–2126, 2018.

[Jonschkowski *et al.*, 2018] Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. Differentiable particle filters: End-to-end learning with algorithmic priors. In *RSS*, 2018.

[Kaelbling *et al.*, 1998] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[Kappen *et al.*, 2012] Hilbert J Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *Machine Learning*, 87(2):159–182, 2012.

[Karkus *et al.*, 2017] Peter Karkus, David Hsu, and Wee Sun Lee. QMDP-net: Deep learning for planning under partial observability. In *NeurIPS*, pages 4694–4704, 2017.

[Karkus *et al.*, 2018] Peter Karkus, David Hsu, and Wee Sun Lee. Particle filter networks with application to visual localization. In *CoRL*, 2018.

[Kempinska and Shawe-Taylor, 2017] Kira Kempinska and John Shawe-Taylor. Adversarial sequential Monte Carlo. In *Bayesian Deep Learning (NeurIPS Workshop)*, 2017.

[Kingma and Ba, 2015] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[Kurniawati and Yadav, 2016] Hanna Kurniawati and Vinay Yadav. An online POMDP solver for uncertainty planning in dynamic environment. In *Robotics Research*, pages 611–629. 2016.

[Levine and Koltun, 2013] Sergey Levine and Vladlen Koltun. Variational policy search via trajectory optimization. In *NeurIPS*, pages 207–215, 2013.

[Levine, 2018] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[Littman *et al.*, 1995] Michael L Littman, Anthony R Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *ICML*, 1995.

[Maddison *et al.*, 2017] Chris J Maddison, John Lawson, George Tucker, Nicolas Heess, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Teh. Filtering variational objectives. In *NeurIPS*, pages 6573–6583, 2017.

[Naesseth *et al.*, 2018] Christian A Naesseth, Scott W Linderman, Rajesh Ranganath, and David M Blei. Variational sequential Monte Carlo. In *AISTATS*, 2018.

[Papadimitriou and Tsitsiklis, 1987] Christos H Papadimitriou and John N Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[Piche *et al.*, 2018] Alexandre Piche, Valentin Thomas, Cyril Ibrahim, Yoshua Bengio, and Chris Pal. Probabilistic planning with sequential Monte Carlo methods. In *ICLR*, 2018.

[Platt Jr *et al.*, 2010] Robert Platt Jr, Russ Tedrake, Leslie Kaelbling, and Tomas Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *RSS*, 2010.

[Ross *et al.*, 2008] Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayes-adaptive POMDPs. In *NeurIPS*, pages 1225–1232, 2008.

[Seiler *et al.*, 2015] Konstantin M Seiler, Hanna Kurniawati, and Surya PN Singh. An online and approximate solver for POMDPs with continuous action space. In *ICRA*, pages 2290–2297, 2015.

[Silver and Veness, 2010] David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. In *NeurIPS*, pages 2164–2172, 2010.

[Somani *et al.*, 2013] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. DESPOT: Online POMDP planning with regularization. In *NeurIPS*, pages 1772–1780, 2013.

[Sunberg and Kochenderfer, 2018] Zachary N Sunberg and Mykel J Kochenderfer. Online algorithms for POMDPs with continuous state, action, and observation spaces. In *ICAPS*, 2018.

[Todorov *et al.*, 2012] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IROS*, pages 5026–5033, 2012.

[Todorov, 2008] Emanuel Todorov. General duality between optimal control and estimation. In *CDC*, pages 4286–4292, 2008.

[Toussaint, 2009] Marc Toussaint. Robot trajectory optimization using approximate inference. In *ICML*, pages 1049–1056, 2009.

[Zhu *et al.*, 2018] Pengfei Zhu, Xin Li, Pascal Poupart, and Guanghui Miao. On improving deep reinforcement learning for POMDPs. *arXiv preprint arXiv:1804.06309*, 2018.