DUAL SEQUENTIAL MONTE CARLO: TUNNELING FILTERING AND PLANNING IN CONTINUOUS POMDPS

Anonymous authors
Paper under double-blind review

ABSTRACT

We present the DualSMC network that solves continuous POMDPs by learning belief representations and then leveraging them for planning. It is based on the fact that filtering, i.e. state estimation, and planning can be viewed as two related sequential Monte Carlo processes, with one in the belief space and the other in the future planning trajectory space. In particular, we first introduce a novel particle filter network that makes better use of the adversarial relationship between the proposer model and the observation model. We then introduce a new planning algorithm over the belief representations, which learns uncertainty-dependent policies. We allow these two parts to be trained jointly with each other. We testify the effectiveness of our approach on three continuous control and planning tasks: the floor positioning, the 3D light-dark navigation, and a modified Reacher task.

1 INTRODUCTION

Partially observable Markov Decision Processes (POMDPs) formulate reinforcement learning problems where the robot's instant observation is insufficient for optimal decision making (Kaelbling et al., 1998). POMDPs can easily become intractable in moderately large discrete space, let alone in continuous domains (Papadimitriou & Tsitsiklis, 1987). As a result, sampling-based methods are typically employed. For example, Monte Carlo tree search (MCTS) methods have shown success in relatively large POMDPs by constructing a search tree of history based on rollout simulations (Silver & Veness, 2010; Seiler et al., 2015; Sunberg & Kochenderfer, 2018). Cross-entropy methods (CEM) (De Boer et al., 2005) update a distribution over policies based on sampled trajectories and have achieved promising results in Decentralized POMDPs (Oliehoek et al., 2008; Omidshafiei et al., 2016). Despite their effectiveness, MCTS methods often require a black-box simulator to generate planning trajectory and thus limiting their success to environments with known dynamics. CEM usually assumes all distributions are Gaussian and therefore is restricted to unimodal planning.

On the other hand, in the context of deep reinforcement learning, approximate solutions for solving POMDPs often directly encode past history with a recurrent neural network (RNN) and performs model-free planning on the latent representations of the RNN (Hausknecht & Stone, 2015; Zhu et al., 2018). Recent work further reduces the burden on the RNN by training an internal generative model for approximate inference of a latent belief (Igl et al., 2018). By doing end-to-end training on the neural architecture, the resulting models can solve complex POMDPs with visual inputs. Despite being simple and generic, these methods lose the ability to incorporate useful prior knowledge like the state formulation as both the planning is completely based on the latent states of RNN. Moreover, whenever these methods fail to perform well, it is difficult to analyze which part causes the failure.

In this work, we present the dual sequential Monte Carlo (DualSMC) model that aims to solve continuous POMDPs with complex unknown dynamics and high dimensional observations, while preserving the interpretability. In particular, DualSMC consists of two coupled inference processes: one for belief estimation over states, and the other for multi-modal density estimation over the optimal future trajectories. To connect the two parts, we feed top particles and the weighted mean estimate from the first SMC, i.e. an adversarial particle filter, as the belief representation to the second one. From there, the second SMC explicitly takes uncertainty into consideration and does multi-modal planning. Note that the learned dynamics is efficiently *shared* between these two parts. The overall pipeline of our algorithm is summarized in Figure 1.

We evaluate our model on three continuous POMDP tasks: the floor-positioning task for explanatory purpose, the 3D light-dark navigation task simulated by DeepMind Lab (Beattie et al., 2016) with rich visual inputs, and a control task in a modified Mujoco (Todorov et al., 2012) environment. Our method consistently outperforms the baseline methods.

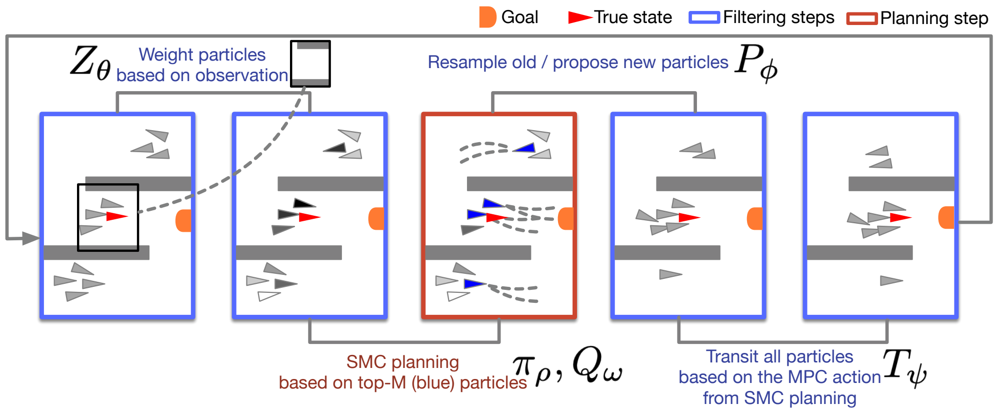

Goal
True state
Filtering steps
Planning step
Z_θ
Weight particles based on observation
Resample old / propose new particles
P_ϕ
SMC planning based on top-M (blue) particles
π_ρ, Q_ω
Transit all particles based on the MPC action from SMC planning
T_ψ

Figure 1: The pipeline of DualSMC. The planner and filter are linked via belief representatives.

2 PRELIMINARIES

In this section, we provide a brief review of the key concepts for the background of this work.

Continuous POMDPs. A continuous POMDP can be 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 underlying continuous state, action and observation spaces. We denote $s_t \in \mathcal{S}$ as the underlying state at time t. When the robot 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 robot 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 robot'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. In general, the optimal policy has to take the entire history into consideration, which can grow exponentially in time steps. Instead of remembering the entire history, classical methods often maintain a belief over possible states and gradually filter out the true state. Denote $\text{bel}(s_t) \triangleq p(s_t|o_{\leq t}, a_{<t})$, then the belief is updated according to $\text{bel}(s_{t+1}) = \eta \int \text{bel}(s_t)\mathcal{Z}(o_{t+1}|s_{t+1})\mathcal{T}(s_{t+1}|s_t, a_t)ds_t$, where η is a normalization factor.

Particle filter (PF). A particle filter uses a set of particles $\{(s_t^{(k)}, w_t^{(k)})\}_{k=1}^{K}$ with $\sum_{k=1}^{K} w_t^{(k)} = 1$, to approximate the belief distribution $\text{bel}(s_t)$. With this approximation scheme, belief update reduces to updating individual particles, $s_{t+1}^{(k)} \sim \mathcal{T}(\cdot|s_t^{(k)}, a_t)$ and $w_{t+1}^{(k)} \propto \mathcal{Z}(o_{t+1}|s_{t+1}^{(k)})w_t^{(k)}$. Particle filter benefits from its flexibility to approximate any distribution. In practice, when the true dynamics $\mathcal{T}$ and $\mathcal{Z}$ are not known a priori, we can use parametrized functions $T_\psi(\cdot)$ and $Z_\theta(\cdot)$ to approximate their corresponding counterparts. Despite its simplicity and efficiency, particle filter can suffer from the particle degeneracy problem that most of the probability mass concentrates on a few particles. A well-known solution is particle resampling, which bootstraps new particles of equal weights from the old ones. Recent advances in particle filters (Karkus et al., 2018b; Jonschkowski et al., 2018) adopt neural networks as the parametrized transition and observation functions and make the filtering process differentiable. By doing so, particle filters can be more easily applied to problems that have complex and high-dimensional observations like images. While previous work focuses on belief space filtering, we take one step further and propose a planning method.

Sequential Monte Carlo planning (SMCP). The task of planning can also be regarded as an inference problem, provided that the likelihood of any future trajectory is proportional to its expected cumulative rewards. This idea is connected to the *control as inference* framework (Todorov, 2008; Toussaint, 2009; Kappen et al., 2012), where control problems are solved by forming a probabilistic graphical model. Specifically, if we denote O_t as the optimality variable and define $p(O_t = 1) \propto \exp(\mathcal{R}(s_t, a_t))$ as the probability of time step t being optimal, the optimal plan then corresponds to the maximum a posterior estimate conditioned on the optimality of all future steps. We can solve this inference problem again with an SMC. For a complete derivation, please refer to Levine (2018) and Piche et al. (2018). Notably, Piche et al. (2018) first uses the sequential Monte Carlo for planning (SMCP) in Markov decision processes. We extend the idea to partially observable domains.

3 METHOD

In this section, we present DualSMC, which consists of two interrelated sequential Monte Carlo processes. We first introduce a more sampling-efficient adversarial particle filter for learning the belief representation, then an uncertainty-aware planning algorithm based on these representations.

3.1 Adversarial Particle Filtering

Similar to the architecture in Jonschkowski et al. (2018), our differentiable PF contains three neural modules, an observation model $Z_\theta(o_t|s_t^{(k)})$ that weights each particle given the current observation, a proposer $P_\phi(s_{\text{new}}^{(k)}|o_t, \epsilon_P)$ for suggesting new probable particles, and a stochastic transition model $T_\psi(s_t|s_{t-1}, a_{t-1}, \epsilon_T)$ that mimics the environment dynamics. Here, ϵ_P and ϵ_T are Gaussian noise for stochasticity. At time t, we re-sample K' particles ($\{s_{\text{old}}^{(k)}\}_{k=1}^{K'}$) transited from the previous step based on the updated weight and combine them with $(K - K')$ newly proposed particles ($\{s_{\text{new}}^{(k)}\}_{k=K'+1}^{K}$)[*]. Naturally, P_ϕ is trained by regressing the proposed state to the true state.

For successful POMDPs planning, the state estimation particle filter had better be really sampling efficient and provide a belief representation based on the most plausible states for the downstream planner. We observe that P_ϕ and Z_θ are opposite yet dependent on each other. Following the intuition that leveraging this adversarial relationship would enhance both parts and thus help to narrow down the belief space of interest, we propose the adversarial particle filter. 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_{real} as the real joint distribution over s, o, Z_θ and P_ϕ play the following two-player minimax game with function $F(Z_\theta, P_\phi)$:

$$\begin{aligned}\min_\phi \max_\theta F(Z_\theta, P_\phi) = \mathbb{E}_{s,o\sim p_{\text{real}}(\cdot)} \big[\log Z_\theta(o|s) + \mathbb{E}_{s\sim s_{\text{old}}^{(k)}} \log(1 - Z_\theta(o|s)) \\ + \mathbb{E}_{\epsilon_P\sim\mathcal{N}(0,I)} \log(1 - Z_\theta(o|P_\phi(o, \epsilon_P))) \big] .\end{aligned} \tag{1}$$

3.2 DualSMC Planning with Belief Representations

A straightforward solution to POMDP planning is to train the planning module separately from the filtering module. At inference time, plans are made based on sampled individual particles from the state filter. We call this planning algorithm *particle-independent* SMC planning (PI-SMCP) and use it as a baseline method. More details about PI-SMCP can be found in Appendix A. Nevertheless, PI-SMCP does not perceive the state uncertainty. By contrast, the proposed DualSMC explicitly considers the belief distribution by planning directly on an approximated belief representation, i.e. a combination of the top candidates from the state filter as well as the weighted mean estimate.

Algorithm 1 Overall DualSMC filtering and planning method

```
1:  $\{s_1^{(k)} \sim \text{Priori}(s_1)\}_{k=1}^K$, $\{w_1^{(k)} = 1\}_{k=1}^K$        ▷ Define initial filtering particles
2:  for $t = 1 : L$ do                                                           ▷ At each control time step
3:      $\{w_t^{(k)} \propto w_{t-1}^{(k)} \cdot Z_\theta(s_t^{(k)}, o_t)\}_{k=1}^K$                 ▷ Update particle weights
4:      $\overline{\text{bel}}_t = \sum_k w_t^{(k)} s_t^{(k)}$                                 ▷ Compute mean belief state
5:      $\{\tilde{s}_t^{(m)}, \tilde{w}_t^{(m)}\}_{m=1}^M = \text{Top-M}(\{s_t^{(k)}, w_t^{(k)}\}_{k=1}^K)$, w.r.t. $\{w_t^{(k)}\}_{k=1}^K$   ▷ Take top-$M$ particles
6:      $a_t =$ DualSMC-P$(\overline{\text{bel}}_t, \{\tilde{s}_t^{(m)}, \tilde{w}_t^{(m)}\}_{m=1}^M; \pi_\rho, Q_\omega)$   ▷ Perform planning (Alg 2)
7:      $o_{t+1}, r_t \sim p_{\text{env}}(a_t)$                                     ▷ Update the environment
8:      $\{s_t^{(k)}\}_{k=1}^{K'} \sim \text{Multinomial}(\{s_t^{(k)}\}_{k=1}^K)$, w.r.t. $\{w_t^{(k)}\}_{k=1}^K$   ▷ Resample particle states
9:      $\{s_t^{(k)} \sim P_\phi(o_t)\}_{k=K'+1}^K$, $\{w_t^{(k)} = 1\}_{k=1}^K$              ▷ Propose new particles
10:     $\{s_{t+1}^{(k)} \sim T_\psi(s_t^{(k)}, a_t)\}_{k=1}^K$                        ▷ Predict next-step particles
11:     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 training buffer
12:     Sample a batch from the buffer and update parameters $(\rho, \omega, \theta, \psi, \phi)$   ▷ Train all modules
13: end for
```

We present the details of DualSMC in Algorithm 1. At time step t, when a new observation comes, we first use Z_θ to update the particle weights (**line 3 Alg 1**), and then perform the planning algorithm in Algorithm 2. We duplicate the top-M particles and the mean belief state N times as the root states of N planning trajectories (**line 1-2 Alg 2**). Different from the previous SMCP (Piche et al., 2018) method under full observations, the policy network π_ρ perceives the belief representations and predicts an action based on the top-M particle states as well as the mean belief state (**line 4 Alg**

[*]Depending on the task, we can keep K' constant or make $(K - K')$ follow an exponential decay.

Algorithm 2 DualSMC planning with filtered belief representations

Input: mean belief state $\overline{\text{bel}}_t$, top-M filtering particles $\{\tilde{s}_t^{(m)}, \tilde{w}_t^{(m)}\}_{m=1}^M$, policy network π_ρ, value network Q_ω, planning start time t, planning horizon H

```
1: $\{\tilde{w}_t^{(m)}\}_{m=1}^M = \text{Normalize}(\{\tilde{w}_t^{(m)}\}_{m=1}^M)$    ▷ Normalize top-$M$ filtering particle weights
2: $\{\{\hat{s}_t^{(m)(n)} = \tilde{s}_t^{(m)}\}_{m=1}^M, \hat{w}_{t-1}^{(n)} = 1, \overline{\text{bel}}_t^{(n)} = \overline{\text{bel}}_t\}_{n=1}^N$    ▷ Duplicate initial states set $N$ times
3: for $i = t : t+H$ do    ▷ At each planning time step
4:     $\{a_i^{(n)} \sim \pi_\rho(\{\hat{s}_i^{(m)(n)}\}_{m=1}^M, \overline{\text{bel}}_i^{(n)})\}_{n=1}^N$    ▷ Sample $N$ actions
5:     $\{\hat{s}_{i+1}^{(m)(n)}, r_i^{(m)(n)} \sim T_\psi(\hat{s}_i^{(m)(n)}, a_i^{(n)})\}_{m=1,n=1}^{M,N}$    ▷ Predict next-step states and rewards
6:     $\{\overline{\text{bel}}_{i+1}^{(n)} \sim T_\psi(\overline{\text{bel}}_i^{(n)}, a_i^{(n)})\}_{n=1}^N$    ▷ Predict next-step mean belief state
7:     $\{\hat{w}_i^{(n)} \propto \hat{w}_{i-1}^{(n)} \cdot \exp\left(\sum_m \tilde{w}_t^{(m)} A^{(m)(n)}\right)\}_{n=1}^N$    ▷ Update weights with Equation 2
8:     $\{x_i^{(n)} = (\{\hat{s}_{i+1}^{(m)(n)}, \hat{s}_i^{(m)(n)}\}_{m=1}^M, \overline{\text{bel}}_{i+1}^{(n)}, a_i^{(n)})\}_{n=1}^N$    ▷ Assemble planning trajectories
9:     $\{x_{t:i}^{(n)}\}_{n=1}^N \sim \text{Multinomial}(\{x_{t:i}^{(n)}\}_{n=1}^N)$, w.r.t. $\{\hat{w}_i^{(n)}\}_{n=1}^N$    ▷ Resample trajectories
10:    $\{\hat{w}_i^{(n)} = 1\}_{n=1}^N$
11: end for
12: Select $a_t$ = first action of $x_{t:t+H}^{(n)}$, where $n \sim \text{Uniform}(1, \ldots, N)$    ▷ Return an action
```

2). We then perform N actions to $M \times N$ states and use T_ψ to predict the next states and rewards (**line 5 Alg 2**). Since future observations $o_{>t}$ are not available at current time step, we approximate $\overline{\text{bel}}_{i>t}^{(n)}$ by re-using T_ψ[†] (**line 6 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 7 Alg 2**). Here, we introduce an alternative advantage formulation that is equivalent to the one used in Piche et al. (2018)

$$A^{(m)(n)} = \text{TD}_{i-1}^{(m)(n)} - \log \pi_\rho(a_i^{(n)} | \{\hat{s}_i^{(m)(n)}\}_{m=1}^M, \overline{\text{bel}}_i^{(n)}), \tag{2}$$

where $\text{TD}_{i-1}^{(m)(n)} = Q_\omega(\hat{s}_i^{(m)(n)}, a_i^{(n)}) - Q_\omega(\hat{s}_{i-1}^{(m)(n)}, a_{i-1}^{(n)}) + r_{i-1}^{(m)(n)}$. 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. We emphasize that our formulation is much simpler. Because our formulation only requires a learned Q function and more importantly, it prevents us from estimating the expectation of the value function V. We leave the full derivation in Appendix B.2.

At the end of each planning time step, we apply the sequential importance resampling (SIR) over N planning trajectories (**line 9-10 Alg 2**). When the planning horizon is reached, where $i = t + H$, we sample one planning trajectory and feed its first action to the environment (**line 7 Alg 1**). We then go back to the filtering part and update the belief representations by resampling, proposing, and predicting the next-step particle states (**line 8-10 Alg 1**). Lastly, we train all modules of DualSMC, including the policy network, the critic network as well as the adversarial particle filtering networks (**line 11-12 Alg 1**). We add details of each network component to Appendix C.

4 Experiment

4.1 Floor Positioning

A robot localizes itself and approaches a target region on a 2D plane. It knows the distances to the nearest walls: $o_t = (d_{x-}, d_{x+}, d_{y-}, d_{y+})_t$, but does not know the world coordinates (s_x, s_y). It starts from a random location and is headed to different regions according to different floors. To reach the correct target, the robot has to learn to reduce the uncertainty about s_y. The action is defined as $a = (\Delta s_x, \Delta s_y)$ with a maximum magnitude of 0.05. Only at training time, a reward of 100 is given at the end of each episode if the robot reaches the correct target region. Implementations details of DualSMC can be found in Appendix D.1.

Qualitative results. Figure 2(a) and 2(b) are results by applying the standard SMCP (Piche et al., 2018) to the top-1 particle state. We use different particle filters (PF) for these two baseline models.

[†]Like QMDP (Littman et al., 1995), DualSMC assumes the uncertainty disappears on the next time step and performs model-based planning.

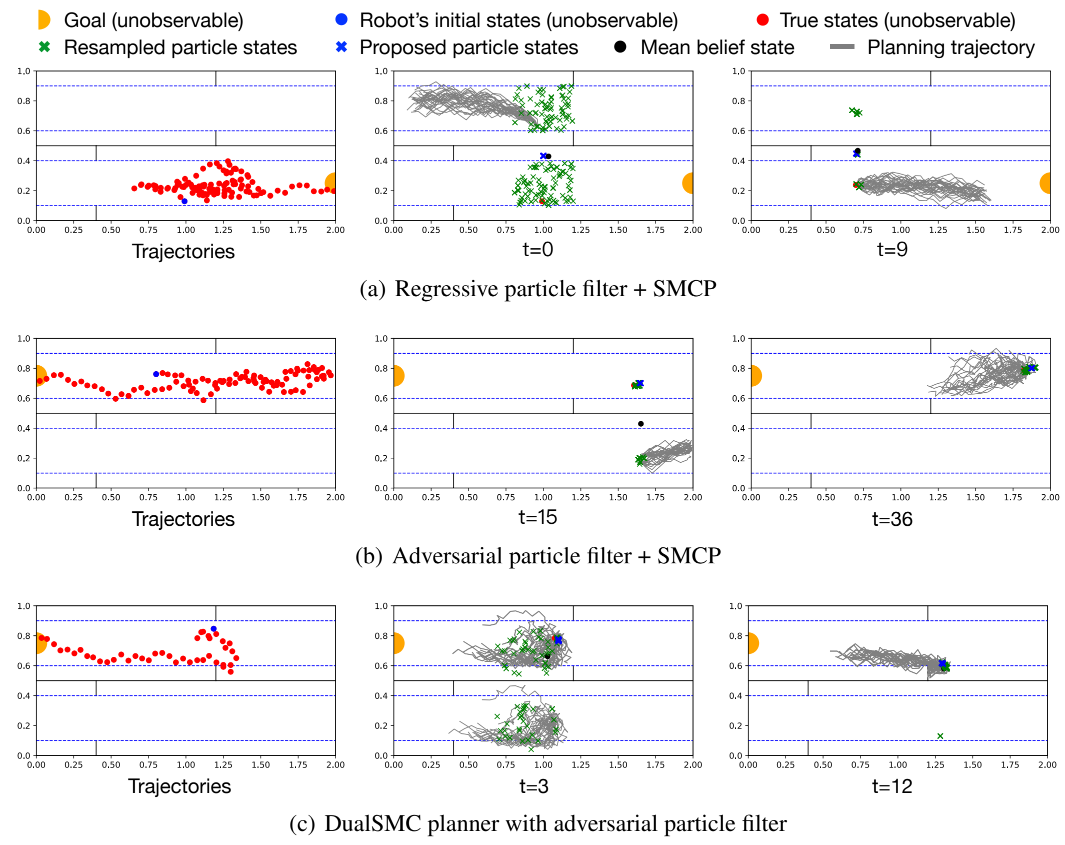


Figure 2: The left column shows the actual trajectories of the robot, and other columns are its planning trajectories at different times. The robot can better localize itself stepping across dashed blue lines.

Table 1: Floor positioning results averaged over 1,000 test episodes, including the success rate, the mean value and the standard deviation of numbers of steps.

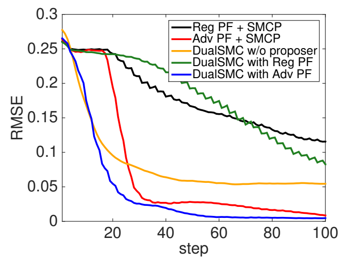
Method	Success rate	# Steps	Std
LSTM filter + SMCP	23.5%	149.1	81.2
DVRL (Igl et al., 2018)	38.3%	162.4	19.9
Regressive PF (top-1) + SMCP	25.0%	107.9	69.8
Adversarial PF (top-1) + SMCP	95.0%	73.3	44.0
DualSMC w/o proposer	78.9%	64.9	62.2
DualSMC with regressive PF (MSE loss)	45.7%	121.6	69.2
DualSMC with regressive PF (density loss)	54.0%	109.2	81.8
DualSMC with adversarial PF	**99.4%**	**36.8**	**15.1**

We use a regressive PF [‡] for the first one and an adversarial PF for the second one. As shown by the blue crosses in Figure 2(a), 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, the robot cannot make reasonable decisions. By contrast, as shown in Figure 2(b), training the PF adversarially leads the proposed particles more akin to plausible states. The robot learns an interesting policy: moving right wherever the starting point is, and then bouncing back at the right wall. However, this policy is clearly not optimal, as it does not consider the uncertainty of the belief state. Figure 2(c) shows the results by DualSMC. We have three findings. First, the robot learns a policy to localize itself quickly and then approach the target area with converged belief states, so that the robot can finish the task successfully with fewer steps. Second, the robot learns the multi-modal distributions of the planning trajectories and can move up or down with different probabilities and advantage values. Third, the observation model works well. Once the robot steps across the blue line, the belief states quickly converge to the actual values.

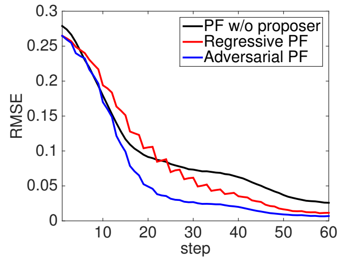
[‡]The term *regressive* here indicates that the proposer of the particle filter takes the *mean square error* between the proposed states and the true states as the training objective function. We may also use the *kernel density estimation* as an alternative objective function.

Quantitative comparisons. Table 1 shows that, first, the DualSMC planning algorithm achieves the highest success rate using the least number of steps. Second, the adversarial PF outperforms other forms of PFs, as well as a deterministic LSTM model (LSTMs are previously used as strong baselines by Karkus et al. (2018b) and Jonschkowski et al. (2018)). DualSMC models with regressive proposers are even worse than one without any proposers, illustrating that the quality of the belief representations is important; erroneous state estimations will harm the final planning results.

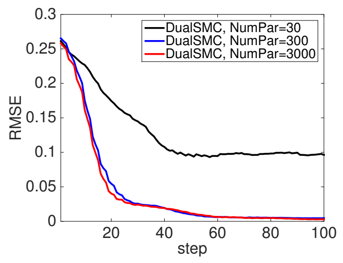
Is the DualSMC filter better than the traditional PF? Given partial observations, an ideal filter should derive a complete distribution of possible states instead of point estimation. Figure 3(a) compares the averaged RMSE between the filtered results of different models and the true states. The adversarial 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 3(b), we train different filters without a planner. All filters follow the same expert trajectories. We can see that the adversarial PF still achieves the best performance. Figure 3(c) explores the effect of different numbers of filtering particles. Using too few particles will deteriorate the filtering performance.



(a) Using different models



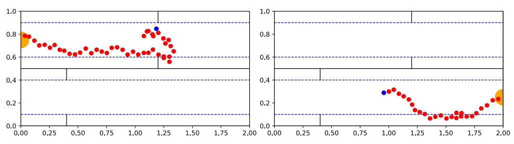
(b) Tracking an expert robot



(c) Changing number of particles

Figure 3: Averaged filtering error as a function of the number of robot steps for floor positioning.

Is the DualSMC planner aware of state uncertainty? In fully observable scenarios, we suppress the filtering part of DualSMC and assume DualSMC plans upon a converged belief on the true state (s_x, s_y). The DualSMC planner makes different decisions to walk straight to the target region (see Figure 4). It performs equally well to the standard SMCP, with an average of 21.3 steps (v.s. 20.7 steps for SMCP) and a 100.0% success rate. We may conclude that DualSMC does not provide policies by remembering them, but by perceiving the distribution of the belief state. We may also conclude that DualSMC trained under POMDPs generalizes well to similar tasks with less uncertainty.


(a) DualSMC with partial observations

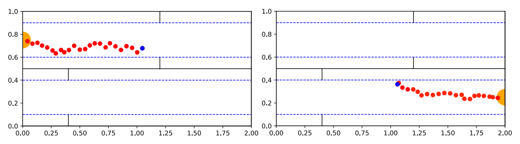

(b) DualSMC with full observations

Figure 4: The DualSMC planner trained under POMDPs generates different polices according to the uncertainty of belief state. For (b), we assign true state values to the top-M particles before planning.

4.2 3D Light-Dark Navigation

We extend the 2D light-dark navigation task (Platt Jr et al., 2010) to a visually rich environment simulated by DeepMind Lab (Beattie et al., 2016). At each episode, the robot is placed randomly and uniformly on one of the 4 platforms at the bottom. Then, the robot's goal is to navigate toward the central cave (marked in orange) while avoiding any of the 4 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 location. 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. Since the moving command in DeepMind Lab is binary (forward/backward), we force the robot to move forward and only let it control its orientation, which we make continuous.

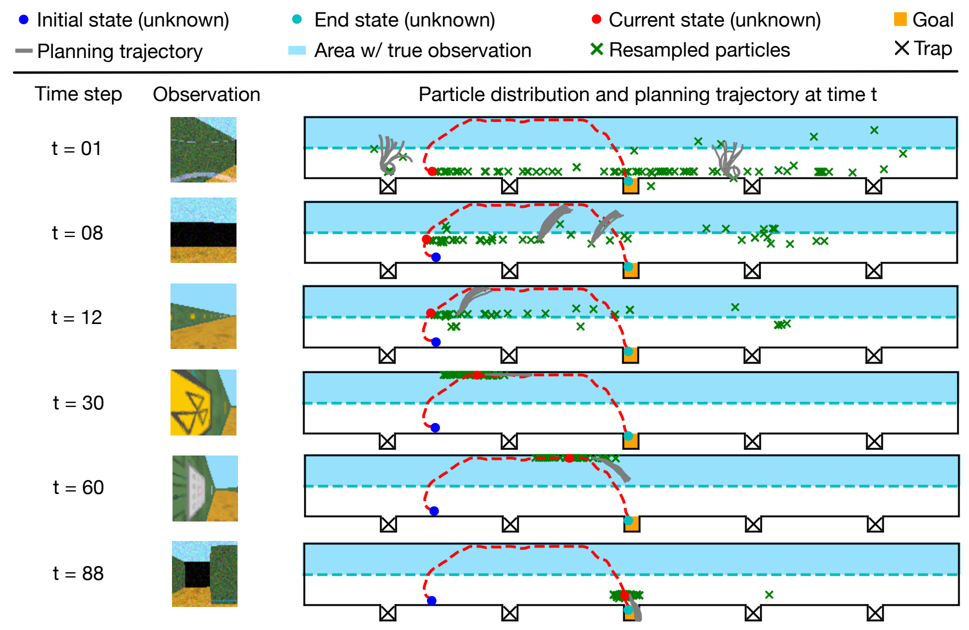


Figure 5: DualSMC learns to go up to the wall of decals first before turning back to the goal. Note that the robot figures out its location at $t = 30$ and where to start turning back at $t = 60$.

The experiment results are summarized in Table 2. By considering the uncertainty, DualSMC methods outperform other baselines and are the only methods that learned to go up and figure out its location first before going directly towards the goal (see Figure 5). However, we notice that DualSMC with the adversarial proposer performs slightly worse than with a proposer trained with density loss. This might be caused by the unstability of the adversarial training. For simplicity, we use the naïve adversarial training method as in the original generative adversarial networks (Goodfellow et al., 2014), which may easily suffer from mode collapse, leading to particle degeneracy in our case. One may potentially improve DualSMC with modern techniques, i.e. Wasserstein distance (Arjovsky et al., 2017), the gradient penalty (Gulrajani et al., 2017), and the spectral normalization (Miyato et al., 2018), to stabilize training or guarantee the existence of a unique Nash equilibrium.

Table 2: Experimental results on the 3D navigation task. Results are averaged over 100 episodes after 2,000 episodes of training. Methods with a filter are trained (tested) with 100 (400) particles.

LSTM + SMCP	59%	85.40
Adversarial PF (top-1) + SMCP	58%	56.11
Adversarial PF (top-3) + PI-SMCP	64%	64.37
DualSMC with regressive PF (MSE loss)	93%	64.32
DualSMC with regressive PF (density loss)	**97%**	73.86
DualSMC with adversarial PF	95%	65.13

4.3 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 $r = ||(r_x, r_y, r_z)||_2 + \epsilon_r$,

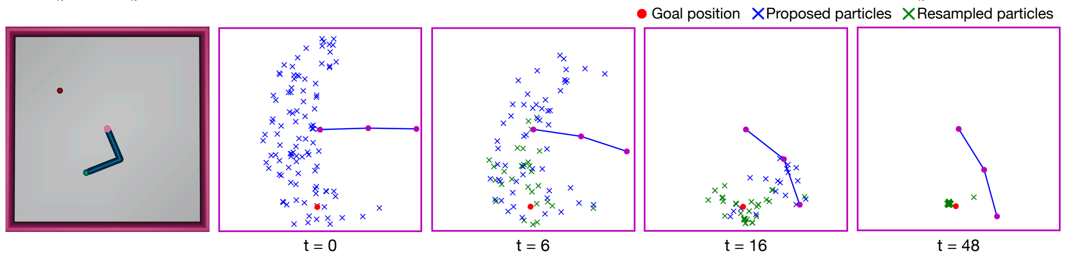


Figure 6: The modified Reacher environment and the goal estimation of DualSMC v.s. time.

where $\epsilon_r \sim \mathcal{N}(0, 0.01)$ is a small noise (notice that r is usually on the scale of 0.1). The resulting 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 6. 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 smoothed training curve of these methods in Figure 7 to showcase the advantage of DualSMC. We truncate the results up to $5{,}000$ episodes since no obvious change in performance is observed from thereon. As we can see, 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.

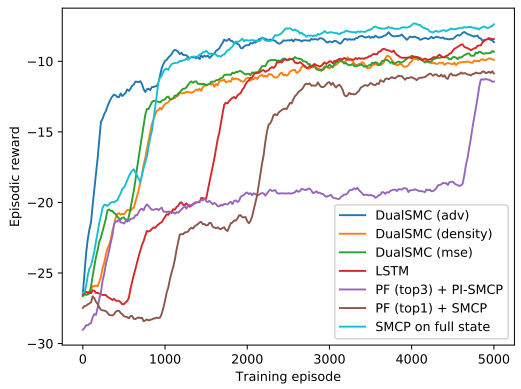


Figure 7: Smoothed training curves on modified Reacher. See experiment details in Appendix D.3.

5 Related Work

Embedding useful algorithmic prior to a reinforcement learning model has been a recent trend because pure model-free approaches can suffer from unstable training and long convergence issues, which are exaggerated in partially observable domains. QMDP-Net (Karkus et al., 2017) extends the idea from value iteration network (VIN) (Tamar et al., 2016) by representing the bellman update in value iteration as convolution and pooling operations in a neural network. A similar idea has been extended to more visually rich domain (Karkus et al., 2018a; Shankar et al., 2016). However, the underlying state and action space still remain simple because of the explicit value iteration procedure during the planning step. The idea of connecting a particle filter with a policy has been addressed in Coquelin et al. (2009). The performance of their policy is only evaluated on the mean particle state and they use finite-difference method to update the policy. Igl et al. (2018) introduced a variational lower bound to improve the original black-box learning of RNN while maintaining a set of particles to address uncertainty. But in their work, the latent particle states are not interpretable and known prior knowledge could not apply. Recently, Karkus et al. (2018b) and Jonschkowski et al. (2018) independently discovered methods to make the conventional particle filter differentiable in terms of neural networks, both showing that end-to-end training improves the filtering performance. Our work extends this idea to planning under uncertainty and we introduce an alternative adversarial proposing strategy to further improve the differentiable filter.

Control as inference methods (Toussaint & Storkey, 2006; Rawlik et al., 2010; Ziebart, 2010; Levine & Koltun, 2013) regard policy search as a probabilistic inference problem by interpreting rewards as the log-likelihood of task fulfillment. It is, therefore, possible to adopt useful statistical tools to solve the control and planning problem by maximizing the likelihood of high reward future trajectory, or estimating the posterior distribution of actions conditioned on the optimality of future trajectories. Levine (2018) provided a comprehensive review of these methods. While previous work mainly focuses on simplified dynamics and simple state/action space, Piche et al. (2018) extended the idea to more complicated task domains by adopting the sequential Monte Carlo approach to select trajectories based on how promising they are, and learning the model-based knowledge necessary for planning in the meantime. All of the above algorithms assume perfect observability and plan in the state space, while our work generalizes the idea to belief space planning.

6 Conclusion

In this paper, we provided a new method named DualSMC to solve continuous POMDPs, which has three advantages. First, it learns plausible belief representations for high-dimensional POMDPs with an adversarial particle filter. Second, it plans future actions by considering the distributions of the learned belief representations. The filter module and the planning module are inter-dependent and jointly trained. 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.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017. 7

C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, J. Schrittwieser, K. Anderson, S. York, M. Cant, A. Cain, A. Bolton, S. Gaffney, H. King, D. Hassabis, S. Legg, and S. Petersen. Deepmind lab. *arXiv:1612.03801*, 2016. 1, 6

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv:1606.01540*, 2016. 7

Pierre-Arnaud Coquelin, Romain Deguest, and Rémi Munos. Particle filter-based policy gradient in pomdps. In *NeurIPS*, 2009. 8

Dan Crisan and Arnaud Doucet. A survey of convergence results on particle filtering methods for practitioners. *IEEE TSP*, 50(3):736–746, 2002. 11

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005. 1

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 7

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *NeurIPS*, 2017. 7

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018. 11

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*, 2015. 1

Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. *arXiv preprint arXiv:1806.02426*, 2018. 1, 5, 8

Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. Differentiable particle filters: End-to-end learning with algorithmic priors. In *RSS*, 2018. 2, 3, 6, 8

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. 1

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

Peter Karkus, David Hsu, and Wee Sun Lee. Qmdp-net: Deep learning for planning under partial observability. In *NeurIPS*, 2017. 8

Peter Karkus, David Hsu, and Wee Sun Lee. Integrating algorithmic planning and deep learning for partially observable navigation. *arXiv preprint arXiv:1807.06696*, 2018a. 8

Peter Karkus, David Hsu, and Wee Sun Lee. Particle filter networks: End-to-end probabilistic localization from visual observations. *arXiv preprint arXiv:1805.08975*, 2018b. 2, 6, 8

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv:1805.00909*, 2018. 2, 8

Sergey Levine and Vladlen Koltun. Variational policy search via trajectory optimization. In *NeurIPS*, 2013. 8

Michael L Littman, Anthony R Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *ICML*, 1995. 4

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018. 7

Frans A Oliehoek, Julian FP Kooij, and Nikos Vlassis. The cross-entropy method for policy search in decentralized pomdps. *Informatica*, 32(4):341–357, 2008. 1

Shayegan Omidshafiei, Ali-Akbar Agha-Mohammadi, Christopher Amato, Shih-Yuan Liu, Jonathan P How, and John Vian. Graph-based cross entropy method for solving multi-robot decentralized pomdps. In *ICRA*, 2016. 1

Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987. 1

Alexandre Piche, Valentin Thomas, Cyril Ibrahim, Yoshua Bengio, and Chris Pal. Probabilistic planning with sequential monte carlo methods. In *ICLR*, 2018. 2, 3, 4, 8, 11, 12

Robert Platt Jr, Russ Tedrake, Leslie Kaelbling, and Tomas Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *RSS*, 2010. 6

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. An approximate inference approach to temporal optimization in optimal control. In *NeurIPS*, 2010. 8

Konstantin M Seiler, Hanna Kurniawati, and Surya PN Singh. An online and approximate solver for pomdps with continuous action space. In *ICRA*, 2015. 1

Tanmay Shankar, Santosha K Dwivedy, and Prithwijit Guha. Reinforcement learning via recurrent convolutional neural networks. In *ICPR*, 2016. 8

David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *NeurIPS*, 2010. 1

Zachary N Sunberg and Mykel J Kochenderfer. Online algorithms for pomdps with continuous state, action, and observation spaces. In *ICASP*, 2018. 1

Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *NeurIPS*, 2016. 8

Emanuel Todorov. General duality between optimal control and estimation. In *CDC*, 2008. 2

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, 2012. 1

Marc Toussaint. Robot trajectory optimization using approximate inference. In *ICML*, 2009. 2

Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes. In *ICML*, 2006. 8

Pengfei Zhu, Xin Li, Pascal Poupart, and Guanghui Miao. On improving deep reinforcement learning for pomdps. *arXiv preprint arXiv:1804.06309*, 2018. 1

Brian D Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, University of Washington, 2010. 8

A Particle-Independent Sequential Monte Carlo Planning

A straightforward baseline method of our planning approach is to compute the future policies of each particle state independently. Similar to DualSMC planning, we initially sample M filtering particles according to their particle weights. Differently, we roll out N future action/state trajectories for each particle independently, following the probabilistic SMCP method (Piche et al., 2018). In the end, there are $M \times N$ future planning trajectories, and we sample one of them according to the importance weights. It is equivalent to the standard SMCP when $M = 1$. We refer this planning algorithm as *particle-independent* SMC planning (PI-SMCP) and use it as an **alternative of Algorithm 2** in **line 6 Algorithm 1**. Details of PI-SMCP are shown in Algorithm 3. We train the PI-SMCP module separately from the filtering network. For the policy network π_ρ, we perform the standard soft actor-critic (SAC) (Haarnoja et al., 2018) on true states. Therefore, one disadvantage of PI-SMCP is that the SAC policy network cannot perceive the uncertainty of the belief representations.

Algorithm 3 Particle-independent SMC planning with filtered belief representations

```
Input: top-$M$ filtering particles $\{\tilde{s}_t^{(m)}, \tilde{w}_t^{(m)}\}_{m=1}^{M}$, policy network $\pi_\rho$, value network $Q_\omega$
1: $\{\hat{s}_t^{(m\times n)} = \tilde{s}_t^{(m)}, \hat{w}_t^{(m\times n)} = \tilde{w}_t^{(m)}\}_{m=1,n=1}^{M,N}$    ▷ Define $M \times N$ initial states for planning
2: $\{\hat{w}_t^{(n)}\}_{n=1}^{M\times N} = \text{Normalize}(\{\tilde{w}_t^{(n)}\}_{n=1}^{M\times N})$    ▷ Normalize weights of planning trajectories
3: for $i = t : t+H$ do    ▷ At each planning time step
4:     $\{a_i^{(n)} \sim \pi_\rho(\hat{s}_i^{(n)})\}_{n=1}^{M\times N}$    ▷ Sample N actions
5:     $\{\hat{s}_{i+1}^{(n)}, r_i^{(n)} \sim T_\psi(\hat{s}_i^{(n)}, a_i^{(n)})\}_{n=1}^{M\times N}$    ▷ Predict next-step states and rewards
6:     $\{\hat{w}_{i+1}^{(n)} \propto \hat{w}_i^{(n)} \cdot \exp(A(\hat{s}_i^{(n)}, a_i^{(n)}, \hat{s}_{i+1}^{(n)}))\}_{n=1}^{M\times N}$    ▷ Update weights, see Eq 2 for $A(\cdot)$
7:     $\{x_i^{(n)} = (\hat{s}_{i+1}^{(n)}, a_i^{(n)}, \hat{s}_i^{(n)})\}_{n=1}^{M\times N}$    ▷ Assemble planning trajectories
8:     $\{x_{t:i}^{(n)}\}_{n=1}^{M\times N} \sim \text{Multinomial}(\{x_{t:i}^{(n)}\}_{n=1}^{M\times N})$, w.r.t.$\{\hat{w}_{i+1}^{(n)}\}_{n=1}^{M\times N}$    ▷ Resample trajectories
9:     $\{\hat{w}_{i+1}^{(n)} = 1\}_{n=1}^{M\times N}$
10: end for
11: Select $a_t$ = first action of $x_{t:t+H}^{(n)}$, where $n \sim \text{Uniform}(1, \dots, M\times N)$    ▷ Return an action
```

B Theoretical Justifications

In this section we provide theoretical justifications for our algorithms.

B.1 Analysis of the POMDPs Filtering Step.

In the following, we use $X_n \xrightarrow{a.s} X$ to denote the sequence of random variables X_n converge to X almost surely. We show if we do not use Top-M selection step, the particle filtering estimator is asymptotically consistent. We remark that Top-M is used for reducing the variance and it may increase the biased and thus M is a hyper-parameter to balance the bias-variance trade-off.

Theorem 1. *Assuming the transition kernel T is Feller and the the likelihood function Z is bounded, continuous and strictly positive, then we have*

$$\lim_{K\to\infty} \overline{bel}_t \xrightarrow{a.s} bel_t.$$

Proof of Theorem 1. Consider the sequence $\{s_t^{(k)}\}_{t=1,k=1}^{L,K}$. Note the evolution of $\{s_t^{(k)}\}_{t=1,k=1}^{L,K}$ (induced by T and Z) forms a particle filtering sequence with resampling. Therefore, we can just apply the Theorem 1 of Crisan & Doucet (2002) to finish the proof. □

B.2 Alternative SMCP Update Formula

We provide the probabilistic graphical model of planning in Figure 8. Similar to the derivation in Appendix A.4 of Piche et al. (2018), we have

$$\begin{aligned}
w_t &= \frac{p(x_{1:t}|O_{1:T})}{q(x_{1:t})} \\
&= \frac{p(x_{1:t-1}|O_{1:T})}{q(x_{1:t-1})}\frac{p(x_t|x_{1:t-1},O_{1:T})}{q(x_t|x_{1:t-1})} \\
&= w_{t-1}\frac{p(x_t|x_{1:t-1},O_{1:T})}{q(x_t|x_{1:t-1})} \\
&= \frac{w_{t-1}}{q(x_t|x_{1:t-1})}\frac{p(x_{1:t}|O_{1:T})}{p(x_{1:t-1}|O_{1:T})} \\
&= \frac{w_{t-1}}{q(x_t|x_{1:t-1})}\frac{p(O_{1:T}|x_{1:t})p(x_{1:t})}{p(O_{1:T}|x_{1:t-1})p(x_{1:t-1})} \\
&= \frac{w_{t-1}}{q(x_t|x_{1:t-1})}\frac{p(O_{1:t-1}|x_{1:t-1})p(x_{1:t})p(O_{t:T}|x_t)}{p(O_{1:t-2}|x_{1:t-2})p(x_{1:t-1})p(O_{t-1:T}|x_{t-1})} \\
&= \frac{w_{t-1}}{q(x_t|x_{1:t-1})}p(x_t|x_{t-1})p(O_{t-1}|x_{t-1})\exp(Q(s_t,a_t)-Q(s_{t-1},a_{t-1})) \\
&= w_{t-1}\frac{p(x_t|x_{t-1})}{q(x_t|x_{t-1})}\exp\Big(Q(s_t,a_t)-Q(s_{t-1},a_{t-1})+r_{t-1}\Big) \\
&= w_{t-1}\frac{p_{env}(s_t|s_{t-1},a_{t-1})}{p_{model}(s_t|s_{t-1},a_{t-1})}\exp\Big(Q(s_t,a_t)-Q(s_{t-1},a_{t-1})+r_{t-1}-\log\pi_\theta(a_t|s_t)\Big)
\end{aligned}$$

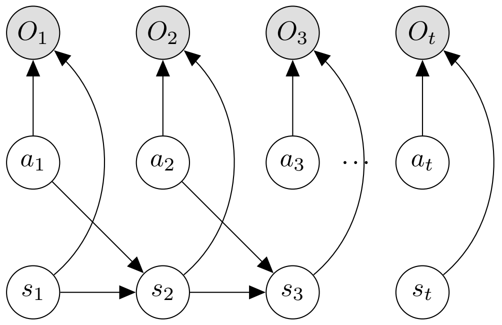


Figure 8: O_t is the observed optimality variable that depends on the latent state s_t and action a_t.

C Network Details

Network details of the DualSMC model for the two planning tasks are summarized in Table 3.

D Implementation Details

D.1 Floor Positioning

We use 100 particles to train the filtering module of DualSMC. Based on the top-3 particles, we roll out 10 steps with 30 planning trajectories. We use Adam optimizer with a 10^{-3} learning rate. At test time, we increase the planning horizon to 20 and the number of filtering particles to 300. All models are trained for 10,000 episodes with a 64 batch size. All results are averaged over 1,000 episodes. The maximum number of steps is 200.

D.2 3D Light-Dark

We use 100 particles to train the filtering module of DualSMC. Based on the top-3 particles, we roll out 5 steps with 10 planning trajectories. We use Adam optimizer with a 0.0003 learning rate. At test

Table 3: Architecture details of the learnable models.

Module	Floor positioning	3D light-dark navigation
Z	$3\times$ fc: $256, 256, 16$	Conv2d: $(3, 3), 16, 2, 1$, ReLU
		Conv2d: $(3, 3), 32, 2, 1$, ReLU, MaxPool(2)
		Conv2d: $(3, 3), 64, 2, 1$, ReLU, Dropout(0.2)
		fc: 128, ReLU (*)
	$2\times$ lstm: $128, 128$	concat: velocity and orientation
	$3\times$ fc: $256, 256, 1$	$5\times$ fc: $128, 128, 128, 128, 1$
P	$3\times$ fc: $256, 256, 64$	same as Z up to (*); fc: 64
	concat: $z(64) \sim \mathrm{Norm}(0, 1)$	concat: orientation, $z(64) \sim \mathrm{Norm}(0, 1)$
	$4\times$ fc: $256, 256, 256, 2$	$3\times$ fc: $128, 128, 2$
T	$4\times$ fc: $256, 256, 256, 4$	action noise: $z(1) \sim \mathrm{Norm}(0, 1)$
		$e = 3\times$ fc(z): $128, 128, 1$
		concat: (state, action + e)
		$3\times$fc: $128, 128, 128, 5$; add to state
Q	$3\times$ fc: $256, 256, 1$	$3\times$ fc: $128, 128, 1$
π	$3\times$ fc: $256, 256, 4$	fc (on top-M): 64; fc (on mean belief): 64; concat; $2\times$ fc: $128, 1$

time, we increase the planning horizon to 15 and the number of filtering particles to 400. All models are trained for 2,000 episodes with a 128 batch size. All results are averaged over 100 episodes. The maximum number of steps is 200.

D.3 Modified Reacher

We use 100 particles to train the filtering module of DualSMC. Based on the top-3 particles, we roll out 1 step (since no multi-modal planning is necessary here) with 10 planning trajectories. We use Adam optimizer with a 0.0003 learning rate. All models are trained for $20{,}000$ episodes with a 256 batch size. As in the original Reacher environment, the maximum number of steps is 50.