Leveraging Fully Observable Policies for Learning under Partial Observability

Anonymous Author(s)
Affiliation
Address
email

Abstract: Reinforcement learning in partially observable domains is challenging due to the lack of observable state information. Thankfully, learning offline in a simulator with such state information is often possible. In particular, we propose a method for partially observable reinforcement learning that uses a fully observable policy (which we call a *state expert*) during training to improve performance. Based on Soft Actor-Critic (SAC), our agent balances performing actions similar to the state expert and getting high returns under partial observability. Our approach can leverage the fully-observable policy for exploration and parts of the domain that are fully observable while still being able to learn under partial observability. On six robotics domains, our method outperforms pure imitation, pure reinforcement learning, the sequential or parallel combination of both types, and a recent state-of-the-art method in the same setting. A successful policy transfer to a physical robot in a manipulation task from pixels shows our approach's practicality in learning interesting policies under partial observability.

Keywords: Partial Observability, Imitation Learning, Fully Observable Expert

1 Introduction

Many real-world robotics control problems feature some degree of partial observability, but policy learning in this setting remains a significant challenge in robotics [1]. While there are many reinforcement learning methods that address partial observability in theory [2, 3, 4, 5], their performance remains questionable in practice in interesting partially observable domains which require long-term memorization and active information gathering [6, 7]. In contrast, the setting of fully observable control has featured the success of many powerful reinforcement learning (RL) algorithms (*e.g.*, [8, 9, 10, 11]). Unfortunately, full observability only holds for a small portion of realistic robotics problems.

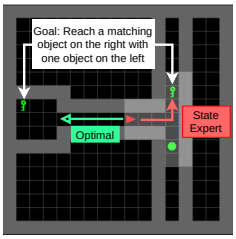


Figure 1: To reach the correct goal object, a state expert takes the red path directly, while a partially observable agent must first take the green path to identify the correct goal object, then take the red path.

In this work, we attempt to leverage good fully observable policies (*state experts*) to help train partially observable policies. Being able to do so would enable us to utilize many powerful methods designed for full observability to solve practical partially observable problems. To employ state information, we rely on the setting of *offline training and online execution*, a successful RL framework where an agent can use "privileged" information such as the state [12, 13, 14, 15] or the belief about the state [6] during offline training, while it is unable to access the privileged information during execution. In this work, the privileged information is not just the state itself but also the state expert. Our setting can be illustrated in a navigation task (Figure 1), which requires an agent to navigate to an unknown goal object on the right, identifiable by an object on the left side. While the optimal behavior under partial observability is to first navigate leftwards to

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

identify the goal object, the state expert is able to move to the goal object directly. Despite being sup-optimal from the partially observable perspective, the state expert has provided experience leading to the goal object, which is potentially useful for both exploration and as a part of the policy needed in the partially observable case after the goal object is identified.

In this work, we propose to perform a form of *cross-observability soft imitation learning* (COSIL), *i.e.*, *softly* projecting the behavior of the fully observable agent into a partially observable agent which still tries to maximize its own performance. Our proposed method is heavily inspired by Soft Actor-Critic (SAC) [10, 16, 17]. The resulting COSIL agent balances between behaviors that imitate the fully observable agent with behaviors that are necessary for partially observable optimality. Experimental results on six robotics domains show that our proposed COSIL method performs significantly better than pure imitation learning, pure reinforcement learning, the sequential or parallel combination of the two types of learning, and other state-of-the-art methods in the same setting. Moreover, a policy with pixel observations learned in simulation is transferred successfully to a physical robot, showing COSIL's robustness and applicability in handling the complexities of the real world even when trained in an approximate simulation.

2 Related Work

Imitation learning methods such as behavioral cloning (BC) [18], and DAgger [19] perform policy learning by minimizing a supervised loss between the actions of the expert and the agent. These agents are trained on a dataset of encountered states and expert actions and are utilized successfully in tasks like autonomous flight in a forest [20, 21] or self-driving cars [22]. Several methods combine reinforcement learning and imitation learning (often from limited expert demonstrations), *e.g.*, DQfD [23], DDPGfD [24], and GAIL [25]. However, these methods are designed for Markov decision processes (MDP), *i.e.*, both the expert and the student can observe the environment state. Closest to our work are ADVISOR [26] and Asymmetric DAgger (A2D) [15], which try to leverage fully observable experts for partially observable tasks. A2D operates in a setting requiring a differentiable and modifiable state expert, trained from scratch (using the environment states) with the student agent, while we assume a given fixed state expert throughout. In the same setting as ours, ADVISOR adaptively changes the weighting per state between the imitation loss and the reinforcement learning loss (see the appendix for a brief description of ADVISOR). However, being our main competitor, it requires training an additional actor using imitation learning to compute the weighting and, in our experiments, performs worse in domains in which the state expert and the optimal student agent differ significantly. Therefore, both ADVISOR and A2D must wait until the auxiliary actor (ADVISOR) and the state expert (A2D) are properly trained before utilizing them to better train the main agent. In contrast, our method can utilize the state expert immediately.

3 Background

In this section, we introduce the background topics required to understand our contributions, *i.e.*, partially observable Markov decision processes, and the Soft Actor-Critic algorithm.

3.1 Partially Observable Markov Decision Processes

A partially observable Markov decision process (POMDP) [27] is a control problem defined by a tuple $(\mathcal{S}, \mathcal{A}, \Omega, p_0, T, R, O, \gamma)$, where $\mathcal{S}$, $\mathcal{A}$, and Ω are respectively the state space, the action space, and the observation space. The initial state is sampled according to the starting state distribution $p_0(s)$, and the next states are sampled according to the stochastic transition function $\Pr(s' \mid s, a) = T(s, a, s')$. The agent does not observe the states directly, but rather receives observations sampled from the observation function $\Pr(o \mid a, s') = O(s', a, o)$. In order to act optimally, a partially observable agent must in general condition its actions on the entire action-observation history $h_t = (a_{<t}, o_{\leq t})$, *i.e.*, all the observable information it has seen so far [28]. Denoting the space of histories

as $\mathcal{H}$, the goal of the partially observable control problem is to find a *history* policy $\pi\colon \mathcal{H} \to \Delta(\mathcal{A})$ which maximizes the expected γ-discounted return $J_\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$.

Although our work focuses on partially observable control, it also involves fully observable agents modeled as a *state* policy $\mu\colon \mathcal{S} \to \Delta(\mathcal{A})$. To differentiate them clearly, we denote history policies as π and state policies as μ.

3.2 Soft Actor-Critic

Soft Actor-Critic (SAC) [10, 16] addresses the fully observable maximum-entropy control problem, *i.e.*, the problem of finding a policy μ which solves a given fully observable control problem while maintaining a high action-entropy when possible. SAC does this by extending the standard RL objective with a weighted entropy term,

$$J_\mu = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t) + \alpha H(\mu(s_t))\right)\right] , \tag{1}$$

where $H(\mu(s)) = \mathbb{E}_{a\sim\mu(s)}\left[-\log \mu(a \mid s)\right]$ is the entropy of policy μ at the given state s, and $\alpha > 0$ is a trade-off coefficient which determines the relative importance between the RL objective and the entropy-maximization objective. The entropy term can be interpreted as a supplementary pseudo-reward given to the agent, which is high for high-entropy policies and low for low-entropy policies. Therefore, the agent will seek not only to maximize the entropy of the policy in the visited states but also to visit states associated with a high-entropy policy.

In practice, SAC is an off-policy learning algorithm which employs a replay buffer containing past transitions $\mathcal{D} = \{(s, a, r, s')_i\}_{i=0}^{N}$. SAC trains a parametric policy model $\mu_\theta\colon \mathcal{S} \to \Delta\mathcal{A}$, and a parametric value model $Q_\phi\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The policy model is trained by maximizing

$$J_\mu(\theta) = \mathbb{E}_{s\sim\mathcal{D}, a\sim\mu_\theta}\left[Q_\phi(s, a) - \alpha \log \mu_\theta(a \mid s)\right] , \tag{2}$$

while the value model is trained to minimize

$$J_Q(\phi) = \frac{1}{2}\mathbb{E}_{s,a\sim\mathcal{D}}\left[\left(R(s, a) + \gamma \mathbb{E}_{s'|s,a}\left[V(s')\right] - Q_\phi(s, a)\right)^2\right] , \tag{3}$$

$$V(s) = \mathbb{E}_{a\sim\mu_\theta(s)}\left[Q_{\bar{\phi}}(s, a) - \alpha \log \mu_\theta(a \mid s)\right] , \tag{4}$$

where $Q_{\bar{\phi}}$ is a frozen target model that is updated at a slower pace than Q_ϕ, to improve stability.

Hyperparameter α plays a central role in SAC, determining how much high-entropy states are preferred to pure rewards. Choosing a reasonable α can be difficult since it is not directly interpretable, and a good value depends dynamically on the current policy's expected returns and entropy. Haarnoja et al. [16] proposed to automatically adjust α by minimizing its own objective,

$$J_\alpha(\alpha) = \alpha \mathbb{E}_{s\sim\mathcal{D}, a\sim\mu_\theta(s)}\left[-\log \mu_\theta(s)\right] - \alpha \bar{H} , \tag{5}$$

where $\bar{H}$ is a given target entropy. In practice, Equation (5) modulates α such that it is increased if the current policy entropy is lower than the target entropy, and vice versa. In contrast to choosing a value of α, choosing a value of $\bar{H}$ is much simpler since it is broadly interpretable as the logarithm of the number of actions that we want the max-entropy policy to consider in an average state.

3.3 Soft Actor-Critic for Partially Observable Control

Although primarily designed for fully observable control problems and state policies μ, SAC can be easily adapted to handle partially observable control problems and history policies π, like most (if not all) model-free learning algorithms. Two main changes need to take place for SAC: First, all appearances of a state s in the equations and models of SAC must be replaced with a respective history h. This also implies the use of a recurrent neural network component in the overall architecture of policy and value models, *e.g.*, an LSTM [29] or a GRU [30]. Second, the replay buffer must be structured in order to contain and extract (truncated) episodes rather than individual transitions.

4 Cross-Observability Soft Imitation Learning

SAC is based on the premise of max-entropy control and is designed to find policies that solve the control task while pertaining to high entropy. This is achieved by extending the standard RL objective with an entropy component which not only pushes the policy model to be more stochastic for any given state but also acts as a pseudo-reward that pushes the policy to visit future states where the policy can be more stochastic. Inspired by this dual effect, we aim to employ a similar technique to exploit fully observable expert knowledge to train a partially observable agent.

Consider a training scenario where a fully observable expert μ is available, *e.g.*, obtained via a planning procedure, a pre-trained model, or a model trained jointly with the partially observable agent. We propose formulating a pseudo-reward for the partially observable agent π based on the expected similarity with the fully observable agent μ, expressed as the following *cross-observability soft imitation learning* (COSIL) objective,

$$J_\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t) - \alpha D(\mu(s_t), \pi(h_t))\right)\right] , \tag{6}$$

where $D(\mu(s), \pi(h))$ is some divergence measure between the action-distributions of μ and π, *e.g.*, the *KL* divergence $D(\mu(s), \pi(h)) = \mathbb{E}_{a\sim\mu(s)}\left[\log \mu(a \mid s) - \log \pi(a \mid h)\right]$, or the *total variation* divergence $D(\mu(s), \pi(h)) = \max_a |\mu(a|s) - \pi(a|h)|$, among other options.

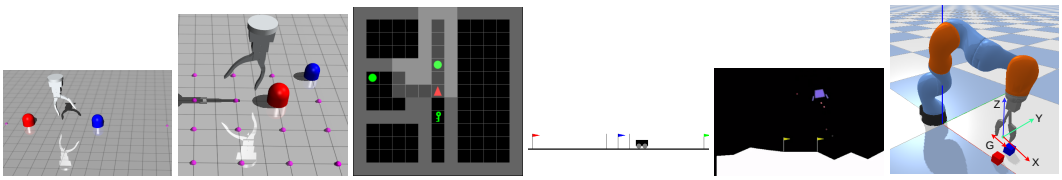
Minimizing the divergence $D(\mu(s), \pi(h))$ can be interpreted as a form of *cross-observability* imitation learning, which tries to project fully observable behaviors as partially observable behaviors. On its own, such a form of imitation learning is known to be sub-optimal. For example, it is not generally possible to replicate the behavior of μ with π. Further, fully observable behaviors are known to be, in general, sub-optimal for partially observable control, *e.g.*, partially observable agents might need to engage in information-gathering behaviors which a fully observable agent would not exhibit. An agent π which strictly minimizes the divergence D would incur an *optimality gap* between its resulting performance and the true optimal performance (see the appendix).

Despite these issues, we argue that applying the divergence objective in a *soft* fashion can be quite beneficial—after all, even the maximum-entropy component of SAC, if applied on its own and without rewards, would result in sub-optimal partially observable policies, and only becomes beneficial when combined with the actual task rewards, which still push the agent towards solving the control problem. First, we argue that the fully observable expert μ is likely to be a good source of exploratory behavior even for the partially observable task. The actions chosen by μ are undoubtedly related to the overall task, and it is likely beneficial for the partially observable agent to explore them thoroughly. Second, there are several partially observable tasks where the agent can and/or must reach low state uncertainty to solve the tasks. Under such low state uncertainty situations, the optimal partially observable and fully observable behaviors overlap strongly, if not perfectly (as long as the state uncertainty is maintained low enough). In such situations, the optimality gap becomes low, and the cross-observability imitation learning task becomes beneficial. Finally, because the soft-imitation task is encoded as a pseudo-reward, the partially observable agent is also pushed to achieve low state uncertainty, which tends to be beneficial for partially observable agents.

Similar to Equation (5), we formulate a separate minimization objective to set α,

$$J_\alpha(\alpha) = \alpha\bar{D} - \alpha\,\mathbb{E}_{h,s\sim\mathcal{D}}\left[D(\mu(s), \pi(h))\right] , \tag{7}$$

which modulates α dynamically in order to maintain an average divergence of $\bar{D}$. Like the target entropy $\bar{H}$ in SAC, $\bar{D}$ is an important hyperparameter that indicates how different the policies π and μ are allowed to be on average. The semantics of $\bar{D}$ depend on the choice of divergence function $D(\pi(h), \mu(s))$, and good values of $\bar{D}$ are likely to be domain-dependent. In practice, we perform a grid search to find the best value. Please see the appendix for the detailed algorithm.



(a) Bumps-1D (b) Bumps-2D (c) Mini-Memory (d) Car-Flag (e) LunarLander (f) Block-Picking

Figure 2: Our experimental domains. The first three domains have discrete actions, the last three feature continuous actions. `Block-Picking` is the only domain with pixel observations.

5 Experiments

We perform experiments on a diverse set of robot and navigation domains with discrete (**D**) and continuous action spaces (**C**), proprioceptive and pixel observations.

5.1 Domains

Below, we only briefly describe the domains and the behavior of the state experts; see the appendix for more details.

(D) Bumps-1D. [6] Two movable homogeneous bumps rest on a table, with a robotics finger moving horizontally above them (Figure 2a). The positions of all entities are randomized starting each episode, but the left-right order between the bumps is maintained. The agent must push the rightmost bump (blue) to the right without disturbing the left bump (red). There are four action combinations: move left or right, each with a compliant or stiff finger. The agent does not know which bump is rightmost – it must infer using the history of the finger's deflected angles and positions while touching both bumps with a compliant finger before pushing right the blue bump with a stiff finger.

(D) Bumps-2D. [6] The same finger is now constrained to move in a plane above two bumps of different sizes (Figure 2b). It must reach the bigger bump (red) and stay there to get the only non-zero reward. However, it is unaware of which bump is bigger and, therefore, must make contact with both bumps at least once, inferring the bumps' relative sizes from the angular displacement of the finger. The state expert, however, knows the position of the bigger bump and can go there directly.

(D) Minigrid-Memory. [31] An agent (red triangle) must go to a matching object located on the right side with an object (a ball or a key) located in a small room on the left side (Figure 2c). The world layout, the type of the matching object, and the starting position of the agent are randomized per episode. The agent might start in a cell where it cannot see the object in the small room; therefore, an optimal agent must first perform information gathering by going/turning left to see the object in the small room. We chose the biggest version of this environment [31] with a size of 17.

(C) Car-Flag. [32] A car must reach the green flag, which can be on either left or right side. The agent normally can only observe the position and velocity of the car, but if being near the blue flag, it can observe the side of the green flag (left/right). The optimal policy will be navigating to the green flag *only* after visiting the blue flag to find the right side. In contrast, the optimal expert will always go left or right towards the green flag, whose position is known to the expert.

(C) LunarLander-P, LunarLander-V. We consider two versions of the classic LunarLander environment [33] where the agent only observes subsets of the full state (see the appendix for more details). The expert, on the other hand, can observe the full state. Masking parts of the state to turn MDPs into POMDPs is common in previous work [34, 3, 35, 5].

(C) Block-Picking. Two homogeneous blocks rest on a table. The red block is fixed to the table and cannot be moved. In contrast, the blue one is movable, and the agent will accomplish the task if it picks the blue block. The agent takes in a top-down colorless depth image; therefore, it must gather information by trying to move both blocks to check for movability. The state expert knows the poses of the movable block exactly; consequently, it can always perform a successful pick.

5.2 Baselines

We compare COSIL against a diverse set of pure imitation learning agents, pure reinforcement learning agents (general and specialized for POMDPs, on-policy, and off-policy), and the combination of both types in several ways. If not described explicitly, all of these agents are *memory-based*.

ADVISOR. ADV-On is the original on-policy ADVISOR [26] that is built off PPO [9]. For a fair comparison with COSIL (off-policy), we implemented an off-policy version of ADVISOR (ADV-Off) using SAC for continuous action spaces [10, 16] and discrete action spaces [17].

DAgger [19] is chosen to represent imitation learning methods instead of a naive behavioral cloning agent, because DAgger often performs better.

TD3 refers to a recurrent version of TD3 [36] for domains with continuous action spaces, implemented in [34]. **SAC** refers to two recurrent versions of SAC for continuous actions [10, 16] and for discrete actions [17], followed the implementation by [34].

BC2SAC is pre-trained with behavioral cloning for $p_{\text{BC}}\%$ of the total training timesteps before being trained with SAC. **BC+SAC** is trained using a weighted combination of behavioral cloning loss $\mathcal{L}_{\text{BC}}$ and RL losses $\mathcal{L}_{\text{RL}}$, *i.e.*, the agent is trained by optimizing $\mathcal{L} = w_{\text{BC}}\mathcal{L}_{\text{BC}} + (1 - w_{\text{BC}})\mathcal{L}_{\text{RL}}$.

VRM is a state-of-the-art off-policy model-based agent [3] for POMDPs with continuous action spaces by combining a recurrent variational dynamic model and a SAC agent.

Random, State Expert. While a random agent takes actions randomly, a state expert is either trained with full state access or is hand-coded using the domain knowledge.

Additional Baselines include recurrent **PPO** [9] implemented in [37], an on-policy specialized method - **DPFRL** [2], and a non-recurrent SAC that uses observations - **Markovian SAC**. We report the performance of these baselines in the appendix.

5.3 Evaluation

Training & Hyperparameter Tuning. All agents are trained using complete episodes. On-policy agents (*i.e.*, DPFRL, PPO, and ADV-On) use 5-10x more samples than other off-policy agents to train. We perform a grid search over relevant hyperparameters of each method (see the appendix for more details), then select the best combinations to report.

Metrics. Depending on each domain, we record either the success rates or the returns of evaluation agents averaged over ten episodes. For SAC-based agents, we turn off exploration during evaluation. The reported results are averages over four seeds with shaded areas denoting one standard deviation with optional smoothing if needed. For Random and State Expert, their performances are averaged over 100 episodes and visualized as horizontal lines.

5.4 Results

The performance of agents are shown in Figure 3 (the performance in `LunarLander-V` is similar to `LunarLander-P` and is reported in the appendix). Overall, COSIL (green) is comparable to ADV-Off in `Bumps-1D`, `LunarLander-P`, `Block-Picking` but better in the remaining three domains. The performance differences are significant in the latter domains, which we argue have larger optimality gaps. ADV-Off does not seem to cope well with such domains, often being outperformed significantly by COSIL. DAgger performs quite well in `LunarLander-P`, suggesting that this domain's optimality gap is small. Our agent can also perform well in that domain, signifying the ability to work in domains with a small optimality gap. Other baselines that do not use the state expert struggle to learn efficiently, especially in `Block-Picking`, where random exploration will have a minuscule chance of picking the correct object. BC2SAC performs poorly across all domains. This way, reinforcement learning often unlearns imitation learning policies due to optimizing RL losses. The poor performance is illustrated by sudden drops whenever reinforcement learning starts. BC+SAC does not seem to perform well either, often not significantly better than DAgger.

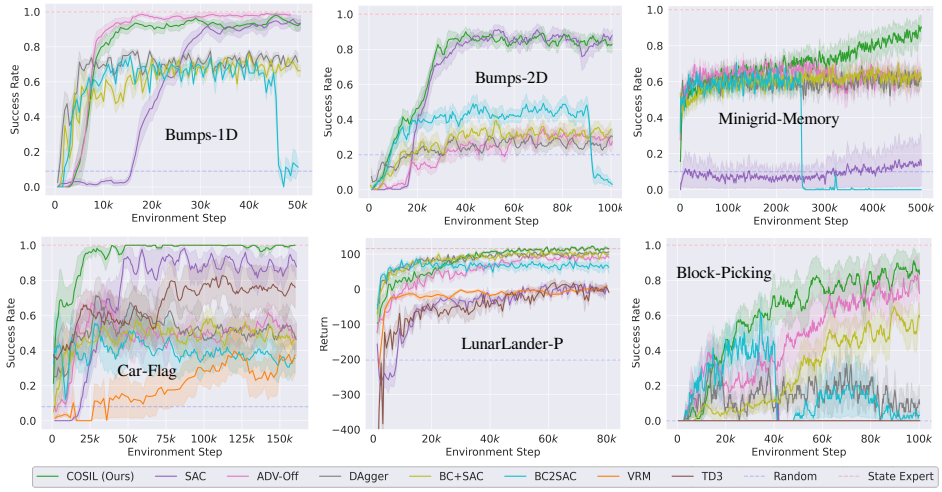


Figure 3: Learning curves of all methods (4 seeds with the shaded areas denoting 1 standard deviation). Note that all agents (except Random and State Expert) are memory-based agents.

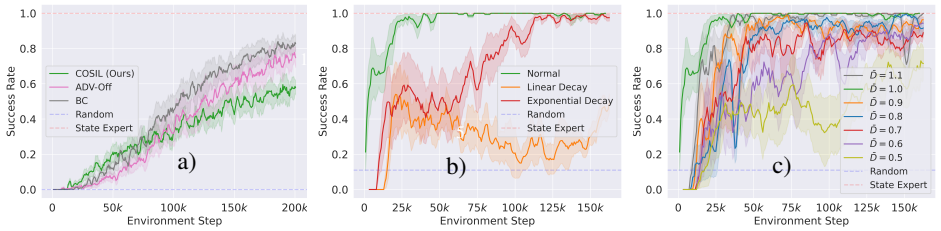


Figure 4: Additional results: a) COSIL in `MiniGrid-Crossing`; Performance of a variant of COSIL in `Car-Flag` when: b) linearly or exponentially decaying $\alpha : 1 \rightarrow 0$; c) varying values of $\bar{D}$.

5.5 Additional Results

Performance in Domains with Small Optimality Gap The performance of COSIL in `LunarLander-P` suggests that it can perform well when the optimality gap is small. When the gap does not exist, *e.g.*, when the state expert is optimal under both full and partial observability, COSIL can also perform relatively well. We demonstrate by a variant of `MiniGrid-Crossing` [31] with a grid size of 25 and 10 crossings (see the appendix). Figure 4a shows that COSIL (with $\bar{D} = 0$) is outperformed by BC (the best agent when no optimality gap exists) and ADV-Off but still performs relatively well. By setting $\bar{D} = 0$, α will be increased to penalize the COSIL agent heavily if it behaves differently from the state expert (see Equation (6)), gradually morphing COSIL into behaving like BC.

Decreasing α over time. Another possible variant of our approach is to reducing $\alpha : 1 \rightarrow 0$ (*e.g.*, linearly or exponentially) over the course of training (instead of automatically adjusting α such that D is kept at $\bar{D}$). The performance of this variant in `Car-Flag` is illustrated in Figure 4b.

Performance with varying $\bar{D}$. We visualize the performance of COSIL in `Car-Flag` with varying values of $\bar{D}$ from 1.1 to 0.5 with a step of 0.1 in Figure 4c.

5.6 Real Robot Experiments

We transfer the policies learned in simulation in `Block-Picking` to a Universal Robot UR5 arm mounted with a Robotiq 2F-85 gripper (Figure 5). We use two cameras: one Occipital Structure Sensor (Cam 1) and one Microsoft Azure Kinect camera (Cam 2), and combine their point clouds to create a depth image by using an orthographic projection at the gripper's position. To better transfer,

during training, we add uniform noise to the positions and the yaw angle of the two blocks and add Perlin noise [38, 39] to the simulated depth images. We pick the best policies in simulation of ADV-Off and COSIL, roll out ten episodes for each, and count the number of successful picks.

During the deployment, when the gripper tries to pick the supposedly immovable block, we must temporarily disable the movements along x, y, g axes (see coordinates in Figure 2f). Without this procedure, even heavily taping or gluing the immovable block will be broken by the powerful gripper. Even worse, when the agent observes that it can move an immovable block, it will keep manipulating that block until the episode times out. The temporal blocking of the movements along specific axes allows us not to use any taping. However, it also leads to cases that the agents never see during training, but both COSIL and ADV-Off seem to generalize.

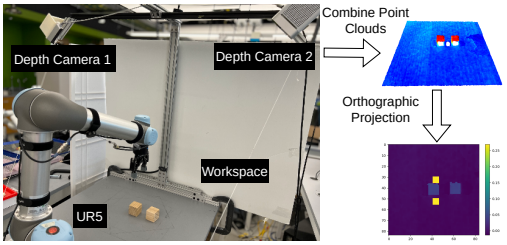


Figure 5: Experimental robot setup.

Method	Simulation	Real-world
ADV-Off	9/10	6/10
COSIL	9/10	8/10

Table 1: Result of transferring policies learned in simulation to the real world in `Block-Picking`.

Table 1 shows the performance comparison of pick successes between COSIL and ADV-Off. In general, our policies are more robust to misalignments of the two blocks in the real world (see the supplementary video for more details), leading to more successful picks.

6 Limitations

COSIL can bias exploration negatively if the state expert is heavily sub-optimal under partial observability, being equal to giving very poor demonstrations. To illustrate, we modify `Bumps-2D` such that staying on top of the bigger bump without visiting the smaller bump in the past will be penalized heavily (-100). Previously, such behavior of the state expert did not incur any penalty. Moreover, to maintain Markovian rewards (*e.g.*, the reward only depends on the last state and action), we add a flag indicating whether the smaller bump was visited to the state and the observation. Figure 6 shows that both COSIL and ADV-Off are outperformed by SAC, with ADV-Off failing to learn completely. For COSIL, setting a high value of $\bar{D}$ might alleviate the situation as COSIL will become closer to a normal SAC agent.

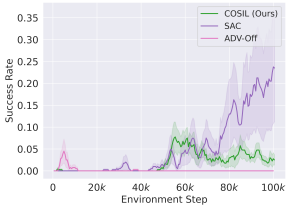


Figure 6: An example when COSIL can be less performant than SAC.

Another limitation is the requirement of constant access to expert actions at every state during training, which might be impractical. One potential fix is using our method with a variant of SAC that learns from limited demonstrations, such as one proposed by Liu et al. [40].

7 Conclusion

This paper tackles POMDPs in robotics from an unconventional angle: leveraging fully observable policies to train better partially observable policies. We perform this task using *cross-observability soft imitation learning*, which balances achieving high partially observable performance with acting similarly to the fully observable expert. With COSIL, the partially observable agent can exploit the fully observable expert both for the purpose of targeted exploration and for the purpose of information-gathering. COSIL outperforms the state-of-the-art method in the same setting in several robotics domains. COSIL also has better performance than other conventional approaches and can learn policies that are successfully transferred to a physical robot.

References

[1] H. Kurniawati. Partially observable markov decision processes and robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1):253–277, 2022. doi:10.1146/annurev-control-042920-092451.

[2] X. Ma, P. Karkus, D. Hsu, W. S. Lee, and N. Ye. Discriminative particle filter reinforcement learning for complex partial observations. *arXiv preprint arXiv:2002.09884*, 2020.

[3] D. Han, K. Doya, and J. Tani. Variational recurrent models for solving partially observable control tasks. *arXiv preprint arXiv:1912.10703*, 2019.

[4] M. Igl, L. Zintgraf, T. A. Le, F. Wood, and S. Whiteson. Deep variational reinforcement learning for pomdps. In *International Conference on Machine Learning*, pages 2117–2126. PMLR, 2018.

[5] L. Meng, R. Gorbet, and D. Kulić. Memory-based deep reinforcement learning for pomdps. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5619–5626. IEEE, 2021.

[6] H. Nguyen, B. Daley, X. Song, C. Amato, and R. Platt. Belief-grounded networks for accelerated robot learning under partial observability. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 1640–1653. PMLR, 16–18 Nov 2021.

[7] H. Nguyen, Z. Yang, A. Baisero, X. Ma, R. Platt, and C. Amato. Hierarchical reinforcement learning under mixed observability. *arXiv preprint arXiv:2204.00898*, 2022.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[10] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[11] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[12] A. Baisero, B. Daley, and C. Amato. Asymmetric dqn for partially observable reinforcement learning. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022.

[13] A. Baisero and C. Amato. Unbiased asymmetric reinforcement learning under partial observability. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 44–52, 2022.

[14] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.

[15] A. Warrington, J. W. Lavington, A. Scibior, M. Schmidt, and F. Wood. Robust asymmetric learning in pomdps. In *International Conference on Machine Learning*, pages 11013–11023. PMLR, 2021.

[16] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[17] P. Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

[18] M. Bain and C. Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[19] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[20] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert. Learning monocular reactive uav control in cluttered natural environments. In *2013 IEEE international conference on robotics and automation*, pages 1765–1772. IEEE, 2013.

[21] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, 2015.

[22] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[23] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[24] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[25] J. Ho and S. Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[26] L. Weihs, U. Jain, I.-J. Liu, J. Salvador, S. Lazebnik, A. Kembhavi, and A. Schwing. Bridging the imitation gap by adaptive insubordination. *Advances in Neural Information Processing Systems*, 34, 2021.

[27] K. J. Astrom. Optimal control of markov decision processes with incomplete state estimation. *J. Math. Anal. Applic.*, 10:174–205, 1965.

[28] S. P. Singh, T. Jaakkola, and M. I. Jordan. Learning without state-estimation in partially observable markovian decision processes. In *Machine Learning Proceedings 1994*, pages 284–292. Elsevier, 1994.

[29] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[30] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[31] M. Chevalier-Boisvert, L. Willems, and S. Pal. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018.

[32] H. Nguyen. Pomdp robot domains. https://github.com/hai-h-nguyen/pomdp-domains, 2021.

[33] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[34] T. Ni, B. Eysenbach, and R. Salakhutdinov. Recurrent model-free rl can be a strong baseline for many pomdps. *arXiv preprint arXiv:2110.05038*, 2021.

[35] T. Gangwani, J. Lehman, Q. Liu, and J. Peng. Learning belief representations for imitation learning in pomdps. In *Uncertainty in Artificial Intelligence*, pages 1061–1071. PMLR, 2020.

[36] S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[37] I. Kostrikov. Pytorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

[38] S. James, A. J. Davison, and E. Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. In *Conference on Robot Learning*, pages 334–343. PMLR, 2017.

[39] K. Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.

[40] G. Liu, L. Zhao, P. Zhang, J. Bian, T. Qin, N. Yu, and T.-Y. Liu. Demonstration actor critic. *Neurocomputing*, 434:194–202, 2021.

[41] J. Achiam. Spinning Up in Deep Reinforcement Learning. 2018.

[42] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.

[43] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[44] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22 (268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

[45] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.