Distilling Motion-Planner Augmented Policies into Visual Control Policies for Robot Manipulation

Abstract: We address the problem of learning complex manipulation tasks in obstructed environments from visual observations. Learning such tasks directly in the visual domain is challenging and sample inefficient due to hard exploration in the presence of obstacles and high-dimensional observation inputs. We propose a framework to learn a visual control policy in a two-step procedure: 1) Learn a policy with long-horizon planning capabilities in state space using motion planning-augmented reinforcement learning; 2) Distilling this policy into the visual domain while removing the dependency on motion planner. In this step, we leverage the exploration and experiences accumulated by the state agent from the first step in an asymmetric actor-critic framework. We evaluate our method on three manipulation tasks in obstructed environments and compare it to various reinforcement and imitation learning baselines. The results indicate that our framework is highly sample-efficient and outperforms state-of-the-art algorithms. We also demonstrate our approach's zero-shot domain transfer capabilities to other environment settings with unseen distractors by coupling our method with domain randomization.

Keywords: Reinforcement Learning, Motion Planning, Robot Manipulation

1 Introduction

Efficiently learning from visual observations has been a long-standing challenge for Reinforcement Learning (RL) algorithms. More so, it becomes even more difficult when solving complex manipulation tasks in cluttered environments, which require precise object-interactions as well as collision-free movements around obstacles. This problem can be tackled by combining the strengths of motion planning (MP) and RL [1, 2, 3] – safe collision-free maneuvers of MP and sophisticated contact-based interactions of RL. However, these methods have practical limitations, such as assuming access to the geometric state of an environment, which is necessary to perform collision checks in MP. Another limitation is the computational cost of motion planners, which makes it difficult to integrate them in a real-time control loop.

In this paper, we propose a two-step approach to learn a visuomotor policy that overcomes these drawbacks. We start by training an agent to solve the task in a simulated environment where access to privileged state information is available. There, we use Motion Planner Augmented Reinforcement Learning (MoPA-RL) [1] to train a policy equipped with a motion planner that can be called to plan collision-free motions and efficiently explore the environment. In the second step, we transfer this state-based policy into the visual domain by distilling the behavior of the motion planner into a direct feedback policy, which is the main focus of this paper. The resulting policy will not need expensive calls to the motion planner and directly select an action based on the visual input, which also makes it suitable for sim-to-real transfer. Directly learning policies from raw pixels has been a challenging problem in RL and even harder for robot learning problems due to the cost and safety concerns involved with real robots. Therefore, learning transferable policies from simulation to real-world robots has significant advantages.

Our visual policy distillation is realized by adopting an asymmetric actor-critic architecture similar to Pinto et al. [4] where the value function is learnt using state information as input whereas the actor directly learns from image observations. This way, our state-based agent's critic and asymmetric critic share the same state space, but have different action spaces. Rather than learning from scratch, we leverage the experience of our state agent and critic to accelerate learning a policy in the visual domain. We achieve this by: 1) Initializing the critic network of our asymmetric agent with the learnt weights of the critic network of MoPA-RL. 2) Leveraging optimal expert trajectories from MoPA-RL and utilizing them for initializing the actor network through pre-training with Behavioral Cloning (BC). While supervised learning with BC does have less training time, it generally fails

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

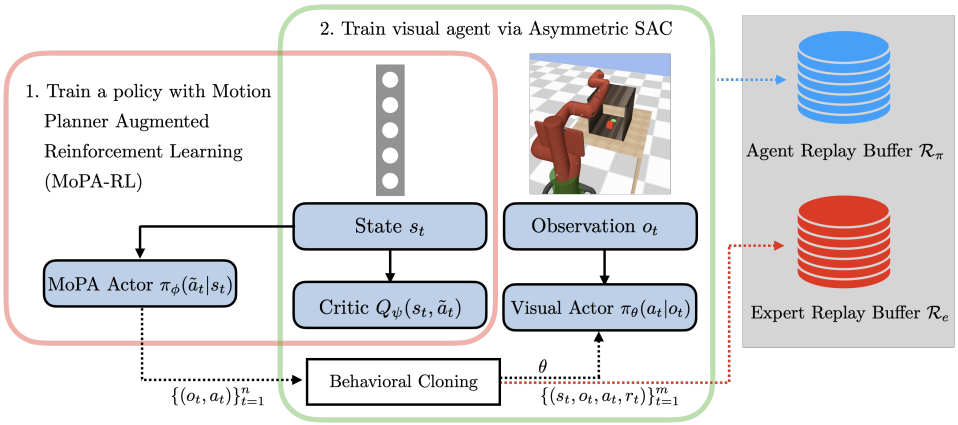


Figure 1: Illustration of our proposed framework where in step 1, we learn a motion planner augmented reinforcement learning (MoPA-RL) policy $\pi_\phi(\tilde{a}_t|s_t)$ in state space. In step 2, we initialize a visual policy $\pi_\theta(a_t|o_t)$ by behavioral cloning MoPA-RL trajectories (o_t, a_t). Moreover, we fill an expert replay buffer $\mathcal{R}_e$ with smooth BC trajectories instead of MoPA-RL trajectories. Then, we train the visual agent with Asymmetric SAC that uses an agent replay buffer $\mathcal{R}_\pi$ along with $\mathcal{R}_e$. Both training steps share the same critic network $Q_\psi(s_t, \tilde{a}_t)$.

in near-collision states due to lack of understanding of appropriate collision thresholds because of unseen environment interactions [5, 6]. We mitigate this issue by training our RL agent on top of BC. We employ the Soft-Actor Critic (SAC) [7] algorithm for training the asymmetric agent.

The main contributions of this work are:

1. A novel framework to learn visual control policies by learning a state-space policy in simulation that uses a motion planner for efficient exploration in cluttered scenarios and subsequently transferring it into the image space while removing the motion planner.
2. An efficient training algorithm for the domain transfer that includes weight initialization, pre-training via BC and fine-tuning with an asymmetric actor-critic algorithm.
3. The evaluation of this approach on three manipulation tasks in obstructed environments and the comparison against state-of-the-art imitation and reinforcement learning baselines.
4. An experiment that demonstrates visual transfer capabilities of our approach by domain randomization of textures, colors and lighting during training.

Our method outperforms state-of-the-art algorithms in terms of success rate, sample efficiency as well as average path length for task completion. Interestingly, our method solves tasks much faster, having a significantly shorter path length compared to all other baselines, including the BC expert trajectories as well as our state-based MoPA-RL agent. This shows that our asymmetric visual agent optimizes its path beyond the expert signals it receives by self-exploration.

2 Related Work

Several recent efforts have been made towards distilling motion planning algorithms into neural motion planners using learning-based methods [8, 9, 10, 11, 5]. Most of these efforts can be broadly categorized into imitation learning (IL) and RL paradigms.

Behavioral cloning, being one of the most common frameworks to learn directly from expert demonstrations, has been used to learn neural network based planners [5, 9, 10]. The works in [12, 9, 11] also adopt supervised learning over expert demonstrations obtained via sampling-based MP, and mostly focus on learning obstacle representations using sensor-measurements like point-clouds or 2D laser scans. However the major drawback of supervised learning approaches is that their performance is limited by the demonstrator's performance itself. Moreover, BC struggles when it encounters data out of the collected data distribution, due to lack of exploration. A common workaround for improving BC is to use DAgger [13], which iteratively gathers expert actions for visited states and

updates the state-action pairs in the training data accordingly. DAgger has also been used in attempts to distill MP into neural networks [5] and for high-level planning [14]. However, its exploitation of the expert for labelling all encountered states can be cumbersome and hard to achieve.

To discover better policies using exploration, various off-policy reinforcement learning methods have been studied for superseding motion planners with neural network policies. They utilise expert demonstrations stored in the replay buffer for guided exploration and succeed to learn policies better than experts [5, 15, 16]. This technique of using expert replay buffers has been derived from popular imitation learning method [17, 18, 19] which were initially introduced on applications outside MP.

Prior works combining RL with expert demonstrations are either focused on unobstructed environments [17, 18, 19, 20] or simpler tasks not involving object manipulation [5]. They also assume fully observable environments and learn policies in state space. Moreover, most offline-RL approaches in sparse reward scenarios are coupled with Hindsight Experience Replay (HER) for goal-conditioned policies. They empirically show that not using HER significantly hurts the agent's performance and might even completely fail at complex tasks [5, 4, 17, 16]. However, using HER is not applicable for composite tasks involving goals associated with multiple sequential steps. This makes HER not suitable for sequential manipulation tasks, where learning from sparse feedback is challenging. Here, we show that our approach successfully performs composite manipulation tasks in obstructed environments without explicitly using HER for tackling sparse rewards. We also learn our policy in the visual domain keeping in mind its applicability for sim-to-sim and sim-to-real transfer.

3 Method

Our goal is to solve complex manipulation tasks in obstructed environments in the visual domain. While MP-based methods can efficiently solve such tasks, they are restricted to state-space learning which is difficult to transfer to real environments. Moreover, sampling-based motion planning incurs significant computational costs that makes it hard to integrate them in real-time controllers. We propose a method which moves from first learning a state-based agent using MP-based actions to subsequently learning a visual policy suitable for zero-shot sim-to-sim and sim-to-real transfers. Our general approach is illustrated in Figure 1. We accelerate visual policy learning by leveraging our state-based agent's experience and therefore increase its sample-efficiency.

3.1 Problem Formulation

We formulate our problem as a continuous state space Markov Decision Process (MDP) for both our state-based as well as asymmetric-visual agent. The MP-augmented MDP $\tilde{M}$ introduced in [1] can be represented as a tuple $(\mathcal{S}, \mathcal{A} \cup \hat{\mathcal{A}}, \tilde{P}, \tilde{R}, \rho_0, \gamma)$ where $\mathcal{S}$ is a set of continuous states, $\mathcal{A}$ is a set of continuous actions in the direct action space and $\hat{\mathcal{A}}$ is a set of continuous actions in the MP-augmented action space for enlarged actions similar to [1]. For the rest of the paper, we refer to this joint action space $\mathcal{A} \cup \hat{\mathcal{A}}$ as $\tilde{\mathcal{A}}$.[1] $\tilde{P} : \mathcal{S} \times \tilde{\mathcal{A}} \times \mathcal{S} \to \mathbb{R}$ defines the transition function, $\tilde{R}$ is the reward function, ρ_0 is the initial state distribution, and γ is the discount factor for $\tilde{M}$.

Next, we represent our asymmetric-visual agent's MDP M by a tuple $(\mathcal{S}, \mathcal{O}, \mathcal{A}, P, R, \rho_0, \gamma)$ similar to [4]. The state space $\mathcal{S}$, initial state distribution ρ_0, and discount factor γ are the same as in the augmented MDP $\tilde{M}$. However, the action space of M is different to $\tilde{M}$ and only consists of the direct action space $\mathcal{A}$, which also leads to a different transition function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ and reward function R. Additionally, we assume to have access to a set of continuous partial observations $\mathcal{O}$ corresponding to the states in $\mathcal{S}$.

At any time step t, the state agent's action distribution is represented by the policy $\pi_\phi(\tilde{a}_t|s_t)$, with it being in state $s_t \in \mathcal{S}$, taking action $\tilde{a}_t \in \tilde{\mathcal{A}}$, where ϕ denotes policy parameters. For our asymmetric visual agent, the actor takes in a partially observable state o_t corresponding to s_t and outputs action $a_t \in \mathcal{A}$, represented by the visual policy $\pi_\theta(a_t|o_t)$ where θ denote the parameters of the visual policy network. The goal of this work is to transfer the behavior of the policy $\pi_\phi(\tilde{a}_t|s_t)$ to the policy $\pi_\theta(a_t|o_t)$, which means to transfer a policy into a different state as well as action space. The learned

[1] MoPA-RL defines the direct action space as $\mathcal{A} = [-\Delta q_{\text{step}}, \Delta q_{\text{step}}]^d$ where Δq_{step} represents the maximum joint displacement. The enlarged action space is $\tilde{\mathcal{A}} = [-\Delta q_{\text{MP}}, \Delta q_{\text{MP}}]^d$ where Δq_{MP} is the motion planner action limit with $\Delta q_{\text{MP}} > \Delta q_{\text{step}}$. For more details, see the definition in Yamada et al. [1].

critic network $\mathcal{Q}_\psi : \mathcal{S} \times \tilde{\mathcal{A}} \rightarrow \mathbb{R}$ is shared between the two MDP's, which is possible since the action space of the asymmetric agent $\mathcal{A}$ is a subset of the action space of the state-based agent $\tilde{\mathcal{A}}$.

At every time step t, the agent receives a reward $r_t = R(s_t, a_t)$, with the state transitioning to s_{t+1}. The goal of the agent is to maximize the expected returns where the return is defined as the discounted sum of rewards $\sum_{t=0}^{T-1} \gamma^t R(s_t, a_t)$, with T depicting the episode horizon.

3.2 Motion Planner-Augmented RL Policy as Expert

In the first step of our approach, we learn a state-based expert policy using the MoPA-RL framework [1]. This framework learns a policy $\pi_\phi(\hat{a}_t|s_t)$ on the augmented MDP $\tilde{M}$ with the Soft Actor Critic (SAC) algorithm. Precisely, the policy predicts actions defined as the joint displacements Δq_t for the robot in each state. Based on the action magnitude, if it lies within the original action space $\mathcal{A}$, this action is directly executed by the controller, similar to any RL policy. This usually aids in performing contact-rich manipulation tasks with high fidelity. When the actions are larger in magnitude, lying in MP-augmented action space $\tilde{\mathcal{A}}$, this often increases the probability of collisions in the presence of obstacles. Therefore, the sampling-based motion planner RRT-Connect [21] is called to realize actions associated with large joint displacements, efficiently planning and exploring obstructed environments. The results in [1] show that this policy structure leads to a more efficient exploration of the state space and a faster convergence of the learning. However, the usage of a motion planner inside an RL policy also has certain weaknesses like high computational costs and the difficulty to transfer it to real scenarios where the environment state is not fully observable. Therefore, we propose a two-step learning procedure in which we start with MoPA-RL and add a subsequent step that transfers the learnt policy into the visual domain while removing its dependency on the MP.

3.3 Learning a Direct Policy in the Visual Domain

The second step of our approach focuses on moving from a state-based MP-augmented policy to an image-based direct policy (without MP) leveraging demonstrations from our MoPA-RL policy as the expert. This stage has two advantages: 1) The learnt policy no longer depends on a sampling-based motion planner for planning collision-free paths in obstructed environments. This makes the policy much faster during learning and inference as MP can be computationally expensive in high-dimensional spaces and slow due to performing collision checks. 2) The learnt policy directly operates on environment images, not requiring state information and thus making it adequate for sim-to-sim and sim-to-real transfers. We now describe different components of our approach in detail.

3.3.1 Trajectory Smoothing using Behavioral Cloning

Our MoPA-RL expert policy is learnt with access to complete information about the environment states at all times and also utilizes MP calls for executing large action steps without collisions. We aim to utilise expert demonstrations from this planner-based policy and distill it into a RL-based visual policy, thereby deducting expensive MP computations as well training the actor with only environment images. We collect multiple transitions t_i into the MoPA-RL demonstration dataset $\mathcal{D}_{\text{mp}} = \{t_1, t_2, t_3, ...\}$, where $t_i = (s_i, o_i, a_i, r_i, s_{i+1})$. Note that in order to learn a vision-based policy in the subsequent step, we render image observations o_i corresponding to each environment state s_i in the simulator during offline data collection.

While MP-based methods can be effective for planning collision-free trajectories in obstacle-prone environments, their paths can be non-smooth [22]. This makes it hard for the robot to learn consistent, naturally smooth motion. Therefore, we first learn a BC agent using observation-action pairs (o_i, a_i) from the dataset $\mathcal{D}_{\text{mp}}$ by minimizing the mean squared error. The BC agent shares the same network architecture and parameters that we use for our subsequent asymmetric actor $\pi_\theta(a_t|o_t)$. We observe that distilling the MoPA-RL trajectories using BC smooths out the rollouts. We then collect multiple rollouts by interacting with the environment using our BC policy and save the refined transitions $(s_i, o_i, a_i, r_i, s_{i+1})$ in an augmented expert replay buffer $\mathcal{R}_e$. The main purpose of using $\mathcal{R}_e$ is to permanently keep hold of expert trajectories during training to facilitate guided exploration. This idea was initially proposed in [18] for discrete action space domains. In our experiments in Section 4, we show that trajectory smoothing for $\mathcal{R}_e$ is crucial for efficiently learning a successful policy.

3.3.2 Learning a Visual Policy with Asymmetric Actor-Critic

To train our RL agent, we adopt an asymmetric actor-critic architecture, similar to [4], where inputs to the critic are full environment states and the actor is learnt with environment images and robot joint states. The choice of this method is motivated by learning transferable visual policies, which do not require state information during inference, and also benefit from access to environment states, when learned in simulation. Our training procedure comprises of the following components:

Initializing actor and critic networks: Initializing network architectures with suitable weights is a promising method for accelerating reinforcement learning [23, 24]. Especially, when doing RL over pixels, meaningful initializations can significantly reduce sample-efficiency [20, 25]. We propose to utilize our state-based agent's experience for initializing our asymmetric visual agent's networks to bring their initial distributions closer to expert signals. The critic $Q_\psi(s_t, a_t)$ is initialized with the weights of the MoPA-RL's critic network. Note that the action space of the two critics is different, but the initialization is still feasible since the action space of MoPA-RL is a superset of the direct action space. For the actor network $\pi_\theta(a_t|o_t)$, we initialize it with the weights of the BC agent that we learned for trajectory smoothing (see Section 3.3.1).

Main training phase: After initializing the actor and critic networks, we initiate the agent's online interactions with the environment. The initial policy rollouts are collected into the agent's replay buffer $\mathcal{R}_\pi$. Further, we adopt a fixed sampling ratio to sample 1:3 transitions from $\mathcal{R}_e$ and $\mathcal{R}_\pi$ respectively, for each training iteration. This ensures that the agent continually improves in an online setting with its own experiences as well as has certain expert transitions to guide exploration. For training our RL agent, we adopt Soft Actor-Critic (SAC) [7], an off-policy model-free RL algorithm for continuous control. We optimize our asymmetric visual policy π_θ by maximizing the objective:

$$J(\pi_\theta) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s_t}, \mathbf{a_t}) \sim \rho_{\pi_\theta}} \left[Q_\psi\left(\mathbf{s}_t, \mathbf{a}_t\right) + \alpha \mathcal{H}\left(\pi_\theta\left(\cdot \mid \mathbf{o}_t\right)\right)\right],$$

where the temperature parameter α is the coefficient of the entropy function $\mathcal{H}$ which regulates the exploration versus exploitation trade-off during learning.

The performance of SAC is known to be sensitive with respect to α [26]. α controls the relative weight of increasing entropy against reward maximization objective. Since our asymmetric agent already starts from experience collected using state-based agent for efficient exploration, we initialize the α parameter to values lower than the final α value obtained from our state-based agent. This is to ensure that with all the prior knowledge in the state-space, the asymmetric visual agent further focuses more on maximizing rewards rather than maximizing entropy for exploration. We describe the implementation details and the choice of hyperparamters in the supplementary.

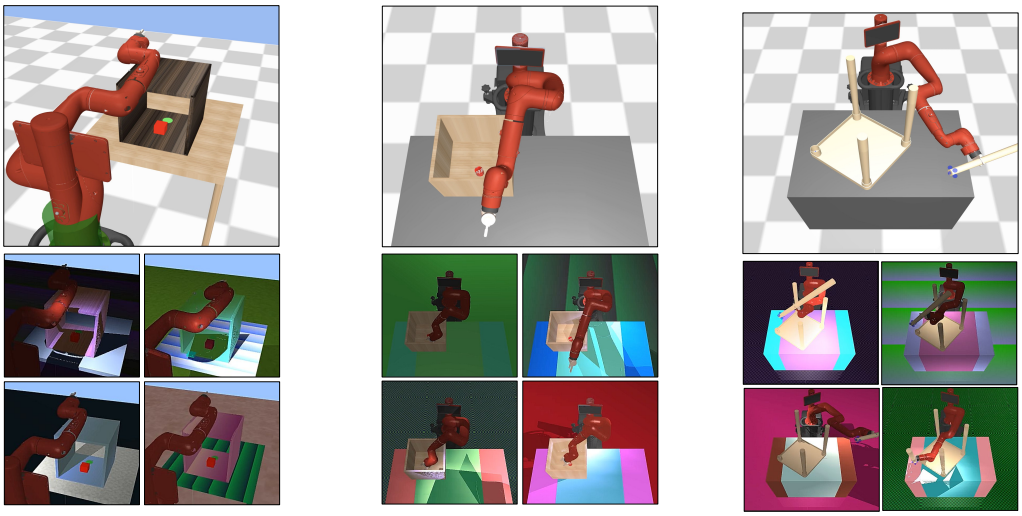
4 Experiments

Our experiments are motivated to answer the following questions: (1) Does IL efficiently learn policies for obstructed environments in image space? Moreover, is naively combining IL with RL sufficient for solving complex tasks? (2) Is learning a visual policy hard using MP with RL? (3) Does our proposed approach efficiently learn a visual policy using prior state-based exploration knowledge? Also, is our learnt visual policy capable of domain transfer and robust to unseen distractors?

4.1 Environments

We evaluate our approach on four obstructed environments, posing hard exploration challenges simulated using the MuJoCo physics engine [27].

- **Sawyer Push:** A Rethink Sawyer robot arm with 7 DoF, initially located outside the box needs to reach an object placed inside the box and push it to the goal location within the box.
- **Sawyer Lift:** The agent needs to reach an object placed inside a box, grasp and lift it outside the box avoiding collisions. The Sawyer's arm is always located outside the box as an initial setup.
- **Sawyer Assembly:** The goal for the Sawyer agent is to assemble the fourth leg of a table into the vacant hole in the presence of the other three legs being already assembled, avoiding any collisions



(a) Sawyer Push (b) Sawyer Lift (c) Sawyer Assembly

Figure 2: Manipulation tasks in obstructed environments from Yamada et al. [1]. (a) *Sawyer Push*: To push the red cube to the goal (green circle). (b) *Sawyer Lift*: Reach the can inside the box and lift it out. (c) *Sawyer Assembly*: Insert the table leg into the hole in the table top. (Top) original environments and (bottom) domain randomization examples.

	Sawyer Push			**Sawyer Lift**			**Sawyer Assembly**		
	ASR	**AEL**	**ADR**	**ASR**	**AEL**	**ADR**	**ASR**	**AEL**	**ADR**
MoPA-RL [1]	98.7	111.3	53.0	94.7	109.7	52.2	99.7	63.2	82.4
BC-Visual	**99.7**	118.1	47.0	68.7	110.2	37.6	94.3	119.3	48.5
Asym. SAC [4]	0.0	250.0	0.0	0.0	250.0	3.27	0.0	250.0	0.0
MoPA-Asym. SAC	0.0	250.0	0.0	0.0	250.0	0.1	0.0	250.0	0.0
CoL [25]	0.0	250.0	0.0	0.0	250.0	3.4	0.0	250.0	0.0
CoL [25] (w BC smoothing)	6.7	165.6	2.4	38.0	89.4	26.4	3.3	145.0	1.6
Ours (w/o BC smoothing)	99.0	37.6	104.3	100.0	42.0	102.3	0.0	250.0	0.0
Ours	**99.7**	**34.5**	**107.7**	**99.7**	**39.2**	**104.5**	**100.0**	**40.3**	**101.7**

Table 1: Average Success rate (ASR), episode length (AEL) and discounted return (ADR) of our method compared to the baselines. MoPA-RL [1] is the expert agent trained with a state-based policy. All baselines below the horizontal line are trained with asymmetric actor-critic for fair comparison. Maximum horizon for successful task completion is 250 across all tasks.

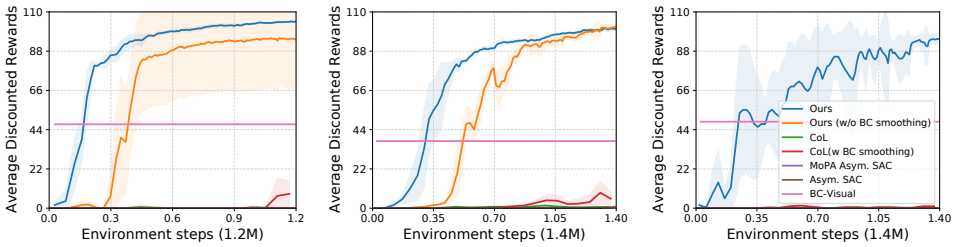
with them. As an initial setup, the leg is already attached to the agent's gripper. This environment is built upon the IKEA furniture assembly environment [28].

Rewards: For *Sawyer Push* and *Sawyer Assembly* environments, we use sparse rewards, where the agent receives a reward proportional to the distance between the end-effector and the object, or the object and the goal position when distance is less than ϵ. For the *Sawyer Lift* tasks, we use reward shaping, similar to [29], where a reward is defined for all three intermediate stages of the task: *reach*, *grasp* and *lift*, based on the distance threshold ϵ. A bonus reward signal of (+150) is received upon successful completion of the task in all the environments.

4.2 Baselines

We compare the following methods as baselines to evaluate the success and merits of our approach:

- **MoPA-RL** [1]: A MP-augmented policy, for learning large displacement actions with MP and smaller actions with RL. This policy also serves as our state-based expert agent. Note that this is the only method in our baselines which learns in the state space.



(a) Sawyer Push (b) Sawyer Lift (c) Sawyer Assembly

Figure 3: Learning curves of our method compared to baselines. Our method solves all 3 tasks with the highest ADR compared to the baselines. Our method w/o BC smoothing learns to solve two tasks with slower convergence compared to our method with BC smoothing but fails on Sawyer Assembly, which shows the significance of trajectory smoothing with BC.

- **Behavioral Cloning (BC-Visual)**: A behavioral cloning policy trained on the image-action pairs collected from the MoPA-RL trajectories using image rendering.
- **Asym. SAC** [4]: An asymmetric actor-critic method, learnt using SAC where the critic is learnt in the state space and the actor is learnt in the image space.
- **MoPA Asym. SAC**: MoPA-RL policy [1] learnt using the asymmetric framework. Note that this method still uses a motion planner with an augmented action space. This method is a direct attempt at learning a visual policy using [1] without the second step of our proposed approach.
- **CoL** [25]: A policy learnt using the Cycle-of-Learning framework, which is a state-of-the-art algorithm for learning from demonstrations (LfD).
- **CoL (w BC Smoothing)** [25]: A policy similar to CoL [25], with BC for trajectory smoothing.
- **Ours (w/o BC Smoothing)**: A policy learnt using our proposed approach described in Section 4 without trajectory smoothing using BC, i.e. we directly using MoPA-RL trajectories in $\mathcal{R}_e$.
- **Ours**: A policy learnt using our proposed approach described in Section 4.

4.3 Evaluation

We compare our proposed approach against baselines on the following evaluation metrics over three different random seeds and 100 unseen episodes:

- **Average Success Rate (ASR):** For all the baselines, we report the average success rate as the mean of the successful episodes evaluated over 3 different seeds, with 100 unseen episodes each.
- **Average Episode Length (AEL):** We also compare our baselines on the basis of the average path length of the successful trajectories.
- **Average Discounted Return (ADR):** We report the average discounted sum of rewards $\sum_{t=0}^{T-1} \gamma^t R(s_t, a_t)$, with T being the episode horizon. An episode completed in more number of steps shall have lower discounted reward, due to the exponentially decaying weight of the discount factor γ with increasing time steps.

Comparison with baselines: Figure 3 compares the learning curves of our method against other baselines. It summarizes the performance using discounted rewards against the number of training steps. As the trend shows, our method is extremely sample-efficient and outperforms all the baselines.

Directly learning a visual policy using Asym. SAC [4] does not begin to learn any of the three tasks. Although Pinto et al. [4] proposed their framework on solving manipulation tasks in a controlled environment, it struggles to learn a visual policy in obstructed environments, where exploration is hard. We also show that MoPA Asym. SAC which is a direct attempt at learning a visual policy while still using MP does not successfully learn to solve the tasks. CoL [25] which has a straightforward combination of BC loss with RL objective for optimizing the actor starts receiving partial rewards after around 1M training steps, much later than our method's convergence but does not achieve a high success rate. Moreover, our method without BC smoothing has a higher variance across different

	Sawyer Push			Sawyer Lift			Sawyer Assembly		
	ASR	**AEL**	**ADR**	**ASR**	**AEL**	**ADR**	**ASR**	**AEL**	**ADR**
Original env.	99.7	39.1	103.1	97.3	57.0	87.4	99.0	58.3	86.9
Env. with distractors	100.0	40.4	102.2	93.0	62.6	84.3	96.7	59.8	87.7

Table 2: Evaluation metrics for domain transfer on original environment (Figure 2) as well as environment with distractors (Figure 4), both unseen during training.

seeds and converges much slower than our method for two tasks. For Sawyer Assembly it does not learn to solve the task at all. This shows the importance of trajectory smoothing in our approach.

As reported in Table 1, BC can achieve decent performance for Sawyer Push and Sawyer Assembly but does not work well for Sawyer Lift environment. However, it has significantly high AEL and low ADR for all tasks, showing its inefficiency to solve the task fast. This is because BC is bound to perform as good as the demonstrations (here collected using MoPA-RL) at its best. In contrast, our method is able to optimize the trajectories beyond expert signals and has significantly lower AEL and higher ADR compared to MoPA-RL, BC and other baselines across all tasks.

Ablation study for weight initialization: We compare our method with and without actor-critic initialization and do not observe any episode success up until 1.2M environment steps for the variant without actor-critic initialization. This shows the importance of initializing the actor as well as the critic for successful training in the visual domain.

4.4 Policy Transfer to Different Domains

In this experiment, we learn our policy along with domain randomization (DR) to verify its domain transfer capabilities. DR is a promising approach for modelling transferable policies using a multitude of variations in the simulated images during training [30, 31], which makes the policy invariant to changes in different domains that are insignificant for the task. For this work, we randomize the simulation environment using different textures, colors, and lighting conditions as shown in Figure 2. We then test our DR-policy on the original environment (see Figure 2) without any randomization, which was never seen by the agent during training. We also simulate a different environment setting with real-world like background distractions such as furniture, walls, floor as illustrated in Figure 4 for evaluating our policy. Apart from the red target cube, which our agent sees during training, we also add two additional cubes (blue and magenta) close to it to evaluate our agent's robustness in solving the task in the presence of distracting objects.

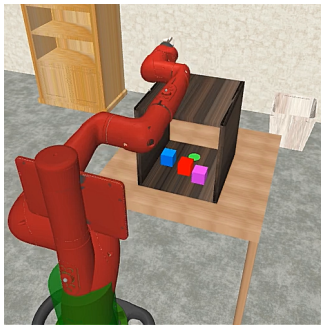


Figure 4: Illustration of our simulated environment for evaluating domain transfer with object distractors such as walls, floor, furniture for Sawyer Push task. We also add 2 distractor cubes (blue, magenta) near the target cube (red) which were not seen during training.

As reported in Table 2, we see that our policy performs significantly well in both testing environments. This illustrates the robustness of our proposed framework in learning transferable visual policies in a sample-efficient manner.

5 Conclusion

In this work, we introduce an efficient method for learning manipulation tasks in obstructed environments in the visual domain. We propose a two-step method to achieve this where in the first step, we learn a policy in the state-space using motion planning with RL. In step two, we use an asymmetric actor-critic framework to accelerate visual policy learning, which removes the motion planner and benefits from the experiences collected by the state-based agent. Since our asymmetric actor does not require state-information, we show that our learnt policy combined with domain randomization is robust under different environment settings with unseen distractors and is competent for zero-shot domain transfers. This work further motivates us to extend our two-step approach from state to visual policy learning on other challenging robotics and RL tasks to ensure its generalizability. We also aim to transfer our visual policy to real-world robots.

References

[1] J. Yamada, Y. Lee, G. Salhotra, K. Pertsch, M. Pflueger, G. S. Sukhatme, J. J. Lim, and P. Englert. Motion planner augmented reinforcement learning for robot manipulation in obstructed environments. In *Conference on Robot Learning*, 2020.

[2] F. Xia, C. Li, R. Martín-Martín, O. Litany, A. Toshev, and S. Savarese. Relmogen: Leveraging motion generation in reinforcement learning for mobile manipulation. In *International Conference on Robotics and Automation*, 2021.

[3] G. Matheron, N. Perrin, and O. Sigaud. Pbcs: Efficient exploration and exploitation using a synergy between reinforcement learning and motion planning. In I. Farkaš, P. Masulli, and S. Wermter, editors, *Artificial Neural Networks and Machine Learning – ICANN 2020*, pages 295–307, Cham, 2020. Springer International Publishing. ISBN 978-3-030-61616-8.

[4] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel. Asymmetric actor critic for image-based robot learning. In *Robotics: Science and Systems*, 2018.

[5] T. Jurgenson and A. Tamar. Harnessing reinforcement learning for neural motion planning. In *Proceedings of Robotics: Science and Systems*, June 2019.

[6] F. Codevilla, E. Santana, A. M. López, and A. Gaidon. Exploring the limitations of behavior cloning for autonomous driving. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9328–9337, 2019.

[7] 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*, 2018.

[8] A. H. Qureshi and M. C. Yip. Deeply informed neural sampling for robot motion planning. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6582–6588, 2018.

[9] A. H. Qureshi, M. J. Bency, and M. C. Yip. Motion planning networks. In *IEEE International Conference on Robotics and Automation*, pages 2118–2124, 2019.

[10] M. J. Bency. *Towards Neural Network Embeddings of Optimal Motion Planners*. PhD thesis, UC San Diego, 2018.

[11] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In *IEEE International Conference on Robotics and Automation*, pages 1527–1533, 2017.

[12] R. Strudel, R. Garcia, J. Carpentier, J. Laumond, I. Laptev, and C. Schmid. Learning obstacle representations for neural motion planning. 2020.

[13] S. Ross, G. J. Gordon, and J. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 2011.

[14] B. Chen, B. Dai, Q. Lin, G. Ye, H. Liu, and L. Song. Learning to plan in high dimensions via neural exploration-exploitation trees. In *International Conference on Learning Representations*, 2020.

[15] H. Ha, J. Xu, and S. Song. Learning a decentralized multi-arm motion planner. In *Conference on Robotic Learning (CoRL)*, 2020.

[16] S. Luo, H. Kasaei, and L. Schomaker. Self-imitation learning by planning. *ArXiv*, abs/2103.13834, 2021.

[17] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. *IEEE International Conference on Robotics and Automation*, pages 6292–6299, 2018.

[18] T. Hester, M. Vecerík, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, G. Dulac-Arnold, J. Agapiou, J. Z. Leibo, and A. Gruslys. Deep q-learning from demonstrations. In *Association for the Advancement of Artificial Intelligence*, 2018.

[19] M. Vecerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. A. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *ArXiv*, abs/1707.08817, 2017.

[20] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[21] J. Kuffner and S. LaValle. Rrt-connect: An efficient approach to single-query path planning. *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, 2:995–1001 vol.2, 2000.

[22] Z. Zhu, E. Schmerling, and M. Pavone. A convex optimization approach to smooth trajectories for motion planning with car-like robots. *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 835–842, 2015.

[23] G. Cruz, Y. Du, and M. E. Taylor. Pre-training neural networks with human demonstrations for deep reinforcement learning. *ArXiv*, abs/1709.04083, 2017.

[24] C. Anderson, M. Lee, and D. L. Elliott. Faster reinforcement learning after pretraining deep networks to predict state dynamics. *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2015.

[25] V. G. Goecks, G. Gremillion, V. Lawhern, J. Valasek, and N. R. Waytowich. Integrating behavior cloning and reinforcement learning for improved performance in sparse reward environments. *In Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS' 20)*, page 465–473, 2020.

[26] Y. Wang and T. Ni. Meta-sac: Auto-tune the entropy temperature of soft actor-critic via metagradient. *ArXiv*, abs/2007.01932, 2020.

[27] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[28] Y. Lee, E. S. Hu, and J. J. Lim. IKEA furniture assembly environment for long-horizon complex manipulation tasks. In *IEEE International Conference on Robotics and Automation*, 2021.

[29] L. Fan, Y. Zhu, J. Zhu, Z. Liu, O. Zeng, A. Gupta, J. Creus-Costa, S. Savarese, and L. Fei-Fei. Surreal: Open-source reinforcement learning framework and robot manipulation benchmark. In *Conference on Robot Learning*, pages 767–782, 2018.

[30] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

[31] M. Sheckells, G. Garimella, S. Mishra, and M. Kobilarov. Using data-driven domain randomization to transfer robust control policies to mobile robots. *2019 International Conference on Robotics and Automation (ICRA)*, pages 3224–3230, 2019.