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

**Anonymous Author(s)**
Affiliation
Address
email

**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 geometric state of an environment, necessary to perform collision checks in MP. Another limitation is the computational cost of MP, 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 visual domain by distilling the behavior of motion planner into a direct feedback policy, which is the main focus of this paper. The resulting policy will not need expensive calls to motion planner and directly select an action based on visual input, which also makes it suitable for sim-to-real transfer. Directly learning from raw pixels has been a challenging problem in RL, even harder for robot learning problems due to the cost and safety concerns 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

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

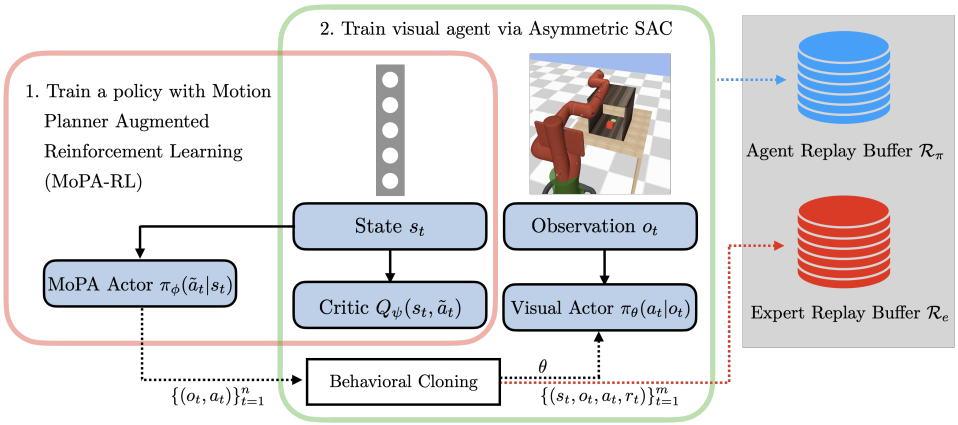



Figure 1: Illustration of our proposed framework where in step 1, we learn a motion planner augmented reinforcement learning (MoPA-RL) policy $\pi_\phi(\tilde{a}_t|s_t)$ in state space. In step 2, we initialize a visual policy $\pi_\theta(a_t|o_t)$ by behavioral cloning MoPA-RL trajectories $(o_t, a_t)$. 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 $Q_\psi(s_t, \tilde{a}_t)$.

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 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, and average path length. 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. As stated in [21], HER is designed to efficiently train a goal-conditioned policy for Multi-goal RL setup, where the policy and critic depend on the state and goal associated with that state. This makes HER not suitable for sequential manipulation tasks, where learning from sparse feedback is challenging without using goal-conditioned policies. Here, we show that our approach successfully performs composite manipulation tasks in obstructed environments without using HER for tackling sparse rewards. We also learn our policy in 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, $\rho_0$ is the initial state distribution, and $\gamma$ 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 $\rho_0$, and discount factor $\gamma$ are the same as in augmented MDP $\tilde{M}$. However, the action space of $M$ is different to $\tilde{M}$ and only consists of direct action space $\mathcal{A}$, leading 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 states in $\mathcal{S}$.

[1] MoPA-RL defines the direct action space as $\mathcal{A} = [-\Delta q_{\text{step}}, \Delta q_{\text{step}}]^d$ where $\Delta q_{\text{step}}$ represents the maximum joint displacement. The enlarged action space is $\tilde{\mathcal{A}} = [-\Delta q_{\text{MP}}, \Delta q_{\text{MP}}]^d$ where $\Delta q_{\text{MP}}$ 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].

Note that the $\mathcal{A}$ is a subset of $\tilde{\mathcal{A}}$ – they have the same dimension but the interval bounds are larger for $\tilde{\mathcal{A}}$ because it also contains motion planner actions. Additionally, the reward and transition functions of both MDPs are the same for direct actions, but only the transition function $\tilde{P}$ handles MP actions.

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 $\phi$ 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 $\theta$ 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 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 $\Delta q_t$ 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 [22] 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 RL-based visual policy, thereby deducting expensive MP computations and training the actor with only environment images. We collect multiple transitions $t_i$ into MoPA-RL demonstration dataset $\mathcal{D}_{\text{mp}} = \{t_1, t_2, t_3, ...\}$, where $t_i = (s_i, o_i, a_i, r_i, s_{i+1})$. The transitions from MoPA-RL are collected in the low-level (direct) action space. 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 $s_i$ 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 [23]. 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 RL [24, 25]. Especially with RL over pixels, meaningful initializations can significantly improve sample-efficiency [20, 26]. We propose to utilize our state-based agent's experience to initialize 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 MoPA-RL's critic weights. Note that the action space of two critics is different, but initialization is still feasible since the action space of MoPA-RL is a superset of direct action space. For the actor network $\pi_\theta(a_t|o_t)$, we initialize it with weights of the BC agent learned for trajectory smoothing (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 $\pi_\theta$ 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 $\alpha$ is the coefficient of the entropy function $\mathcal{H}$ which regulates the exploration versus exploitation trade-off during learning.

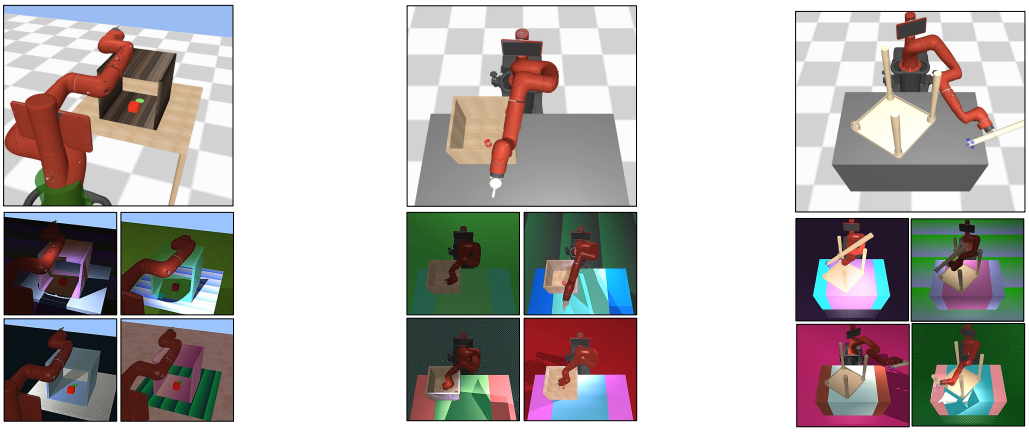
The performance of SAC is known to be sensitive with respect to $\alpha$ [27]. $\alpha$ controls the relative weight of increasing entropy against reward maximization. Since our asymmetric agent already starts from experience collected using state-based agent, we initialize $\alpha$ to values lower than the final $\alpha$ obtained from our state-based agent. This is to ensure that with prior knowledge in state-space, the asymmetric visual agent focuses more on maximizing rewards rather than maximizing entropy for exploration. We describe the implementation details and choice of hyperparameters in 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 three obstructed environments, posing hard exploration challenges simulated using the MuJoCo physics engine [28]. We use a 32x32 image as a visual observation. The state information for each environment is described in Section C.



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

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

| | **Sawyer Push** | | | **Sawyer Lift** | | | **Sawyer Assembly** | | |
|---|---|---|---|---|---|---|---|---|---|
| | **ASR** ↑ | **AEL** ↓ | **ADR** ↑ | **ASR** ↑ | **AEL** ↓ | **ADR** ↑ | **ASR** ↑ | **AEL** ↓ | **ADR** ↑ |
| MoPA-RL | 98.2 | 111.0 | 52.7 | 95.0 | 109.2 | 52.7 | 99.8 | 63.3 | 82.4 |
| BC-Visual | 99.4 | 118.0 | 46.9 | 62.0 | 108.8 | 34.6 | 97.0 | 115.1 | 50.4 |
| Asym. SAC | 0.0 | 250.0 | 0.0 | 0.0 | 250.0 | 0.2 | 0.0 | 250.0 | 3.4 |
| MoPA-Asym. SAC | 0.0 | 250.0 | 0.0 | 0.0 | 250.0 | 0.0 | 0.0 | 250.0 | 0.0 |
| CoL | 0.0 | 250.0 | 0.0 | 29.8 | 173.3 | 9.7 | 0.0 | 250.0 | 0.0 |
| CoL w/ BC smoothing | 0.0 | 250.0 | 0.0 | 0.0 | 250.0 | 5.1 | 0.0 | 250.0 | 0.0 |
| Ours w/o BC smoothing | **100.0** | 34.0 | 108.7 | **99.4** | 43.6 | 100.7 | 0.0 | 250.0 | 0.0 |
| Ours | **100.0** | **32.0** | **110.8** | 99.0 | **42.0** | **101.7** | **100.0** | **61.7** | **84.5** |

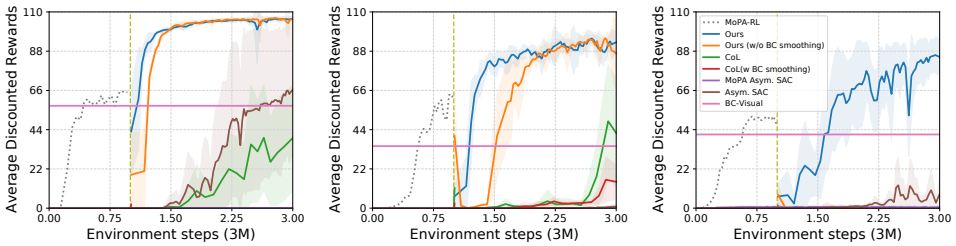
Table 1: Average success rate (ASR), episode length (AEL), and discounted return (ADR) of our method and baselines averaged over five seeds. Each method is evaluated after 3M environment steps. MoPA-RL [1] is the expert agent trained with state-based policy. All baselines below horizontal line are trained with asymmetric actor-critic for fair comparison. Maximum horizon is 250 for all tasks.

- **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 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 [29].

*Rewards:* For *Sawyer Push* and *Assembly* environments, we use sparse rewards, where the agent receives a reward proportional to the distance between the end-effector and object, or the object and goal position when distance is less than $\epsilon$. We use $\epsilon = 0.1$ for Sawyer Push and $\epsilon = 0.3$ for Sawyer Assembly environment while the distance between initial and goal state is around 1.2. An episode is considered successful when the distance between the cube (Sawyer Push) and goal or the peg-head and goal (Sawyer Assembly) is less than 0.05. For *Sawyer Lift* task, we use reward shaping, similar to [30], where a reward is defined for all three intermediate stages of the task: *reach*, *grasp* and *lift* . A bonus reward signal of (+150) is received upon successful task completion in all the environments.

### 4.2 Baselines

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




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

Figure 3: Learning curves of our method compared to baselines. We train MoPA-RL for 1M env steps, followed by 2M steps for baselines requiring MoPA-RL pre-training (total 3M). Other baselines not requiring MoPA-RL are also trained for 3M steps for a fair comparison. For BC, we report training time in Table 8. Our method solves all the tasks with highest ADR. Our method w/o BC smoothing can solve two tasks with slower convergence compared to Ours but fails on Sawyer Assembly.

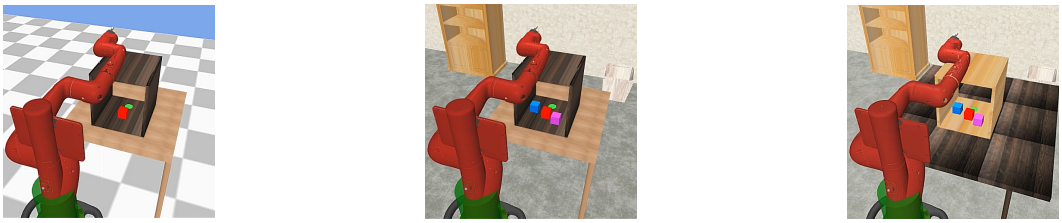
- **MoPA-RL** [1]: A MP-augmented policy, for learning large displacement actions with MP and smaller actions with RL. This policy also serves as our state-based expert agent.
- **Behavioral Cloning (BC-Visual)**: A behavioral cloning policy trained on the image-action pairs collected from the MoPA-RL 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** [26]: 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)** [26]: A policy similar to CoL [26], 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 averaged over five 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 with 100 unseen episodes per random seed.
- **Average Episode Length (AEL):** We compare baselines on average length of successful episodes.
- **Average Discounted Return (ADR):** We report average discounted sum of rewards $\sum_{t=0}^{T-1} \gamma^t R(s_t, a_t)$, with $T$ being episode horizon. An episode completed in more time steps has lower discounted reward, due to exponentially decaying weight of the discount factor ( $\gamma = 0.99$).

*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 far more sample-efficient and outperforms all the baselines. Directly learning a visual policy using Asym. SAC [4] does not completely learn any task. 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, a direct attempt at learning a visual policy while still using MP does not successfully learn to solve the tasks. CoL [26] which has a straightforward combination of BC loss with RL objective for actor optimization receives partial rewards for some tasks after around 1.5-2M 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 seeds and converges slower



(a) Original env. (b) Modified env. 1 (c) Modified env. 2

Figure 4: Illustration of unseen environments for evaluating domain transfer: (a) original environment without randomization, (b) modified env. 1 with distractors such as walls, floor, furniture. We also add 2 distractor cubes (blue, magenta) near the target cube (red) which were not seen during training, and (c) modified env. 2 similar to (b) but with variation in table size and, box and table texture.

| | **Sawyer Push** | | | **Sawyer Lift** | | | **Sawyer Assembly** | | |
|---|---|---|---|---|---|---|---|---|---|
| | **ASR** ↑ | **AEL** ↓ | **ADR** ↑ | **ASR** ↑ | **AEL** ↓ | **ADR** ↑ | **ASR** ↑ | **AEL** ↓ | **ADR** ↑ |
| Original env | 99.7 | 39.1 | 103.1 | 99.3 | 37.3 | 106.5 | 100.0 | 49.7 | 93.4 |
| Modified env 1 | 100.0 | 40.4 | 102.2 | 96.7 | 38.3 | 103 | 100.0 | 47.9 | 94.8 |
| Modified env 2 | 99.3 | 43.4 | 99.04 | 97.3 | 37.0 | 104.7 | 100.0 | 47.7 | 95.0 |

Table 2: Evaluation metrics for domain transfer on unseen environments with distractors (Figure 4.)

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 method. As reported in Table 1, BC achieves decent performance for Sawyer Push and Assembly but does not work well for Lift task. 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 trajectories beyond expert signals and has significantly lower AEL and higher ADR compared to all other baselines for all tasks.

*Ablation study for weight initialization:* We compare our method with and without actor-critic initialization and do not observe any episode success until 1.2M environment steps for the latter. This shows the importance of initializing actor and 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 simulation during training [31, 32], which makes the policy invariant to changes 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 three unseen scenarios, as illustrated in Figure 4 comprising of 1) the original environment without any randomization; 2) modified environment 1, with real-world like background distractions such as furniture, walls, and floor. We also add unseen distractor cube (blue and magenta) near the target (red) cube; 3) modified environment 2, which is similar to the previous, but has additional changes in the texture of the table and box, and different size of the table. As reported in Table 2, we see that our policy performs significantly well in all three testing environments. This illustrates the robustness of our proposed framework in learning transferable visual policies in a sample-efficient manner. We further discuss about the key learning and limitations of our work in Section D.

## 5 Conclusion

In this work, we introduce an efficient method for learning manipulation tasks in obstructed environments in visual domain. We propose a two-step method where in step one, we learn a policy in state-space using motion planning and RL. In step two, we use an asymmetric actor-critic framework to accelerate visual policy learning, which removes the motion planner and benefits from experiences collected by state-based agent. We also show that our visual policy combined with domain randomization is robust under different environment settings with unseen distractors and is competent for zero-shot transfers. This work motivates us to extend our approach on other challenging tasks to ensure generalizability for future work. We also aim to transfer our policy to real 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] M. Andrychowicz, D. Crow, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. In *NIPS*, 2017.

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

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

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.