

## A Additional Ablation Study

We discuss further ablation results regarding: (1) The effect of the entropy coefficient  $\alpha$  on the performance as well as convergence speed for learning our visual policy. (2) The effect of using our proposed weight initialization strategy for the asymmetric actor-critic framework. (3) The effect of behavioral cloning for trajectory smoothing over the motion planner augmented RL trajectories. (4) The dependency of our method on the success of MoPA-RL.

### A.1 Effect of varying the entropy coefficient $\alpha$

In Section 3.2, we discuss the role of tuning the entropy coefficient  $\alpha$  for maintaining the trade-off between entropy maximization as well as reward maximization for the SAC objective. For our visual policy learning using asymmetric actor-critic framework, we mainly focus on the exploration achieved by our state-based agent using MoPA-RL [1]. This helps us leverage our state agent’s experience for providing guided exploration rather than exploring the entire state space again for visual policy learning. Therefore, we focus on the reward maximization objective by choosing a very small value of  $\alpha$ .

Table 3 reports the  $\log(\alpha)$  values for our learned state-agent using MoPA-RL. For accelerating our visual policy learning, we use smaller  $\alpha$  values to further emphasize on reward maximization. In Figure 5, we show that using smaller values of  $\alpha$  assists in faster convergence and improving sample efficiency as compared to using higher  $\alpha$  values. Moreover, smaller  $\alpha$  values consistently converge for all the environments whereas for larger  $\alpha$  values, we can see unstable performance varying across different tasks. We also observe that large values of  $\alpha$  undergo large perturbations during training making learning unstable whereas smaller values of  $\alpha$  remain stable throughout training, as shown in Figure 6.

	Sawyer Push	Sawyer Lift	Sawyer Assembly
$\log(\alpha)$	-0.63	-2.7	-0.29

Table 3: Values of the final  $\log(\alpha)$  after MoPA-RL training for all three environments.

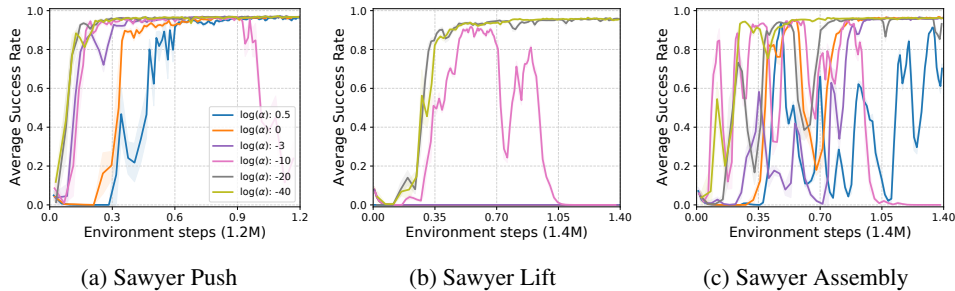


Figure 5: Ablation curves for showing the effect of the entropy coefficient  $\alpha$  for visual policy learning via asymmetric actor-critic framework. As the trends show, smaller  $\alpha$  values lead to a stable and faster training for all three environments.

### A.2 Effect of weight initialization

As discussed in Section 4.3, we note that our weight initialization technique for the actor-critic networks before visual policy learning significantly aids in achieving optimal performance and also facilitates improving learning speed. In Figure 7, we see that without appropriate initialization for the asymmetric critic using our state agent and the visual actor using behavioral cloning, the agent achieves minimal or no success rate across all environments up till 1.2M environment steps. On the other hand, our method with initialization converges to a success rate of  $\approx 1.0$  much earlier than 1.0M environment steps for all three tasks.

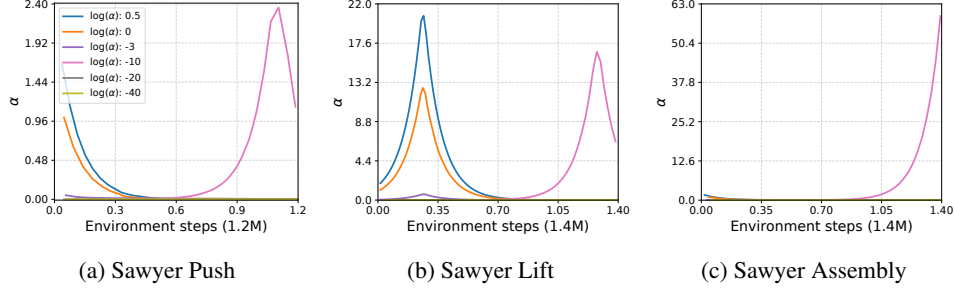


Figure 6: Ablation curves for showing the change in the entropy coefficient  $\alpha$  during training for visual policy learning via asymmetric actor-critic framework. As the trends show, larger  $\alpha$  values fluctuate during training leading to unstable learning, whereas smaller values of  $\alpha$  do not show large variations throughout training for all three environments.

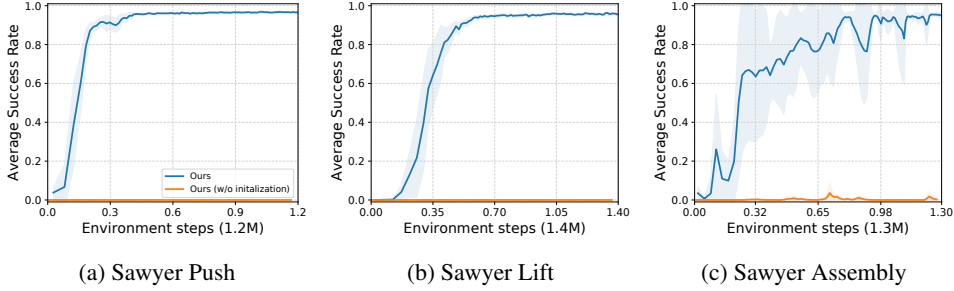


Figure 7: Learning curves comparing our method with and without actor-critic initialization for all three environments. Without initialization, the agent achieves minimal or no success rate, whereas it quickly learns to achieve optimal performance with weight initialization across all the tasks.

### A.3 Behavioral cloning for trajectory smoothing

As mentioned in Section 3.2.1, we use BC to smooth out jittery motion planner augmented trajectories and store them in an augmented expert replay buffer  $\mathcal{R}_e$ . In Figure 8, we qualitatively compare BC and MoPA-RL’s random rollouts with the same start and goal positions. As we can visualize, the BC trajectory path is smoother compared to the trajectory obtained via the MoPA-RL policy. This is indeed significant for a refined transition from a state-based policy to a visual policy in terms of performance as well as convergence speed and sample efficiency (see Figure 3).

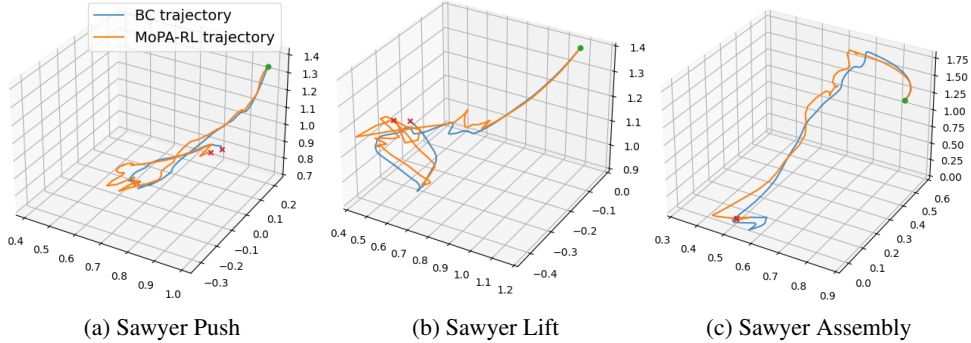


Figure 8: Visualization of end-effector position for a randomly chosen rollout comparing the smoothness of behavioral cloning trajectories with respect to MoPA-RL trajectories with the same start state (marked in green) and final end-effector position (marked in red) for all three tasks.

	Environment Steps					
	0.2M			1.0M		
	ASR $\uparrow$	AEL $\downarrow$	ADR $\uparrow$	ASR $\uparrow$	AEL $\downarrow$	ADR $\uparrow$
MoPA-RL [1]	69.2	102.7	41.0	98.0	105.4	54.5
BC-Visual	93.2	137.4	37.1	98.8	57.4	97.2
Ours	100.0	34.4	108.2	100	32.0	110.8

(a) Sawyer Push

	Environment Steps								
	0.5M			0.52M			1.0M		
	ASR $\uparrow$	AEL $\downarrow$	ADR $\uparrow$	ASR $\uparrow$	AEL $\downarrow$	ADR $\uparrow$	ASR $\uparrow$	AEL $\downarrow$	ADR $\uparrow$
MoPA-RL [1]	41.4	153.8	21.0	60.6	138.3	29.2	95	109.2	52.7
BC-Visual	48	189.4	14.1	55.2	146.6	24.2	63	109.4	34.8
Ours	99.0	42.0	101.6	99.4	42.9	101.0	99.0	42.0	101.7

(b) Sawyer Lift

Table 4: Average success rate (ASR), episode length (AEL), and discounted return (ADR) of our method compared with MoPA-RL and BC using sub-optimal and optimal MoPA-RL models. Even with sub-optimal training, when our state-based agent does not learn to completely solve the task, our method for distillation into a visual policy achieves high performance and successfully solves the task in lesser steps (smaller AEL).

#### A.4 Sub-optimal training for MoPA-RL

To analyze the dependence of our approach on the success of the MoPA-RL training in stage one, we experiment with sub-optimal models of MoPA-RL. We train on lesser environment interactions and then use the critic for weight initialization, followed by our asymmetric visual agent training. We train for the same number of environment steps as Table 1 results, i.e., 2M environment steps. We report the results in Table 4a and Table 4b for Sawyer Push and Sawyer Lift tasks, respectively.

As reported in Table 4a and Table 4b, we observe that although our method benefits from initialization using the state-based agent compared to random initialization, it does not heavily depend on the state-based agent’s success. This shows that even when MoPA-RL cannot completely solve the task, our framework for distilling the state-based policy into the visual policy achieves optimal performance.

## B Implementation Details

### B.1 Network architecture and hyperparameter selection

For training our state-based agent in step one using MoPA-RL [1], we use three layered fully connected neural networks with 256 hidden units and ReLU activation for actor  $\pi_\phi$  and critic  $Q_\psi$  networks and MP implementation similar to [1] using the RRT-Connect algorithm. For our asymmetric agent’s critic, we retain the critic network same as  $Q_\psi$  and the visual actor  $\pi_\theta$  is a 3-layered convolution neural network followed by three fully connected layers with 256 hidden units and Leaky ReLU activation and another set of fully connected layers outputting the mean and the standard deviation of the Gaussian distribution over an action space. The behavioral cloning agent shares the network architecture with the asymmetric visual actor  $\pi_\theta$ . We specify other hyperparameter details for SAC and behavioral cloning training in Table 5, Table 6 and Table 7.

Parameter	Value
Optimizer	Adam
Learning rate	1e-5
Discount factor ( $\gamma$ )	0.99
Replay buffer size	$10^6$
Image size	32x32
Minibatch size	256
Nonlinearity	ReLU
No. of expert trajectories	100
Network update per env. step	1

Table 5: SAC hyperparameters shared across all environments

Parameter	Value
Optimizer	Adam
Learning rate	5e-4
# observation-action pairs	$\approx 1\text{M}$
Train-test split	9:1
Image size	32x32
Minibatch size	512
Nonlinearity	LeakyReLU
Scheduler step size	5
Scheduler decay rate	0.99

Table 6: Behavioral cloning hyperparameters shared across all environments

	Sawyer Push	Sawyer Lift	Sawyer Assembly
Action dimension	7	8	7
Reward Scale	0.8	0.5	1.0

Table 7: Environment-specific parameters for SAC training

## B.2 Choosing the optimal model for behavioral cloning

For behavioral cloning training, we use validation success rate for choosing the optimal model weights rather than using loss on the test dataset. As seen in the Figure 9, we see that a lower error on the test set does not necessarily provide the best validation accuracy. Therefore, we pick the model on the first such epoch which receives 100% validation accuracy. For measuring the validation accuracy after each training epoch, we use 5 episodes each on 6 randomly chosen seeds. We train our BC agent for a total of 140 epochs and select the earliest model with the best validation accuracy for BC trajectory smoothing as well as asymmetric visual actor’s weight initialization. We also report the training time and number of epochs (for optimal convergence) required to account for the additional steps required in learning the BC agent in Table 8.

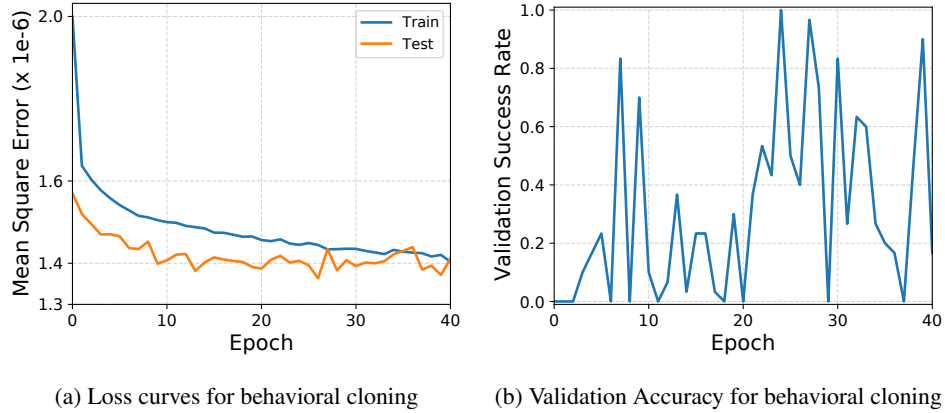


Figure 9: Learning curves for behavioral cloning showing loss on the train set, loss on the test set as well as the validation accuracy per epoch for Sawyer Assembly environment.

## C Domain Randomization

For domain randomization, we randomly sample a variation for the texture, color and lighting conditions for each episode during training. An illustration of the domain randomization samples for

	Sawyer Push	Sawyer Lift	Sawyer Assembly
Number of epochs	12	13	24
Wall-clock time	30	39	120

Table 8: Number of epochs and wall-clock time (in minutes) for training a BC agent used for weight initialization and BC Smoothing.

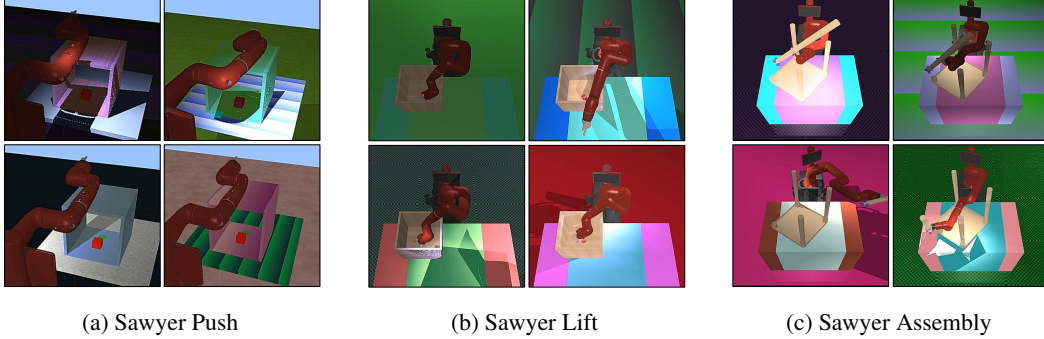


Figure 10: Illustration for random samples of domain randomization for each environment,

each environment is shown in Figure 10. Training our method with domain randomized environments helps in learning robust policies that are capable of transfer to unseen domains and distractor objects as shown in Section Figure 4.5.

## D Environment Details

We simulate all of our 3D environments using MuJoCo physics engine [31]. Subsequently, we describe the observation details for all three environments. We maintain all other specifics regarding the reward functions, success criteria as well as environment initial states same as explained in Yamada et al. [1].

### D.1 Sawyer Push

The task is to reach an object placed inside a box and push it towards the goal region using a Sawyer Arm. We define the positions of the goal, object and the end-effector as  $p_{\text{goal}}$ ,  $p_{\text{obj}}$  and  $p_{\text{eef}}$  respectively.

**Observations.** Each state observation  $s_t$  comprises of the joint state  $(\sin \theta, \cos \theta)$ , angular joint velocity, quaternion end-effector coordinates,  $p_{\text{eef}}$  the position of the object, the position of the goal  $p_{\text{goal}}$ , distance between the end-effector and the object, and the distance between the object and goal position. This acts as an input to the critic in the asymmetric framework.

The visual observation  $o_t$  comprises of the simulated image corresponding to  $s_t$  and the robot joint space information. This is used as an input to the actor in the asymmetric framework for learning the visual policy.

### D.2 Sawyer Lift

For Sawyer Lift, the task is to reach an object placed inside a box, grasp it using the gripper hand and then lift it up above the box height.

**Observations.** The state observation  $s_t$  comprises of the joint state  $(\sin \theta, \cos \theta)$ , angular joint velocity, the position of the object position and quaternion, end-effector coordinates,  $p_{\text{eef}}$ , the position of the goal  $p_{\text{goal}}$ , and distance between the end-effector and the object. This acts as an input to the critic in the asymmetric framework.

For an input to the asymmetric actor for visual policy learning, the visual observation  $o_t$  comprises of the simulated image corresponding to  $s_t$  and the robot joint space information.

### D.3 Sawyer Assembly

In Sawyer Assembly, the task is to assemble the fourth leg of a tabletop in its gripper to its corresponding hole where the other three legs are already in place. This needs to be done while avoiding collisions with the obstructions posed by the other three table legs that are already assembled since the table is free to move under collisions.

**Observations.** The state observation  $s_t$  comprises of the joint state  $(\sin \theta, \cos \theta)$ , angular velocity, the position of the hole, the head and tail positions of the leg in the gripper hand and its quaternion.

The input to the asymmetric actor for visual policy learning comprises of the simulated image  $o_t$  and the robot joint state information.