

## Appendix

### A Simulation Environment

We develop our planar pushing simulation environment using NVIDIA Omniverse Isaac Sim due to its GPU parallelization capabilities, which significantly accelerate the training and data collection process. Fig. 5 shows a visualization of the simulation environment. The rectangular planar workspace has dimensions  $0.6 \text{ m} \times 0.3 \text{ m}$ . Note that we designed this workspace based on the maximum reachability in our robotic hardware set-up. For the manipulated object, we use a cuboid of size  $0.12 \text{ m} \times 0.1 \text{ m} \times 0.07 \text{ m}$ , and for the pusher we use a sphere of radius  $0.013 \text{ m}$ . We enforce the workspace boundaries with respect to the centroids of the pusher and the object. During policy training and data collection, we randomize the dynamics of the environment, including the mass of the manipulated object as well as friction and restitution coefficients of the table, the pusher, and the object. Table 3 summarizes the randomized dynamics parameters and their corresponding sampling distributions.

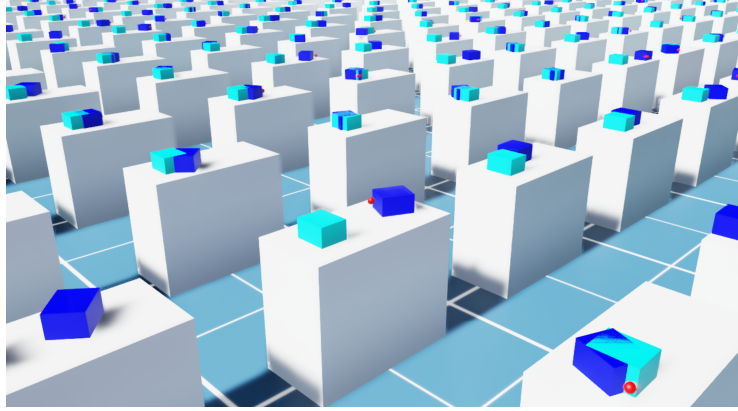


Figure 5: Planar pushing simulation environment in Isaac Sim. The pusher is shown in red, the manipulated object in dark blue, and the target pose in light blue.

Parameter	Distribution
Static friction	$\mathcal{U}(0.3, 0.5)$
Dynamic friction	$\mathcal{U}(0.1, 0.3)$
Restitution	$\mathcal{U}(0.1, 0.7)$
Object mass	$\mathcal{U}(3.0, 3.5) \text{ kg}$

Table 3: Dynamics randomization parameters and corresponding sampling distributions.

### B Additional Training Details

#### B.1 Reinforcement Learning Policies

We process the RL observation by scaling each component to the range  $[-1, 1]$ . In particular, we scale the  $(x, y)$  coordinates using the workspace dimensions and for the object orientation  $\theta$  we use  $\sin(\theta)$  and  $\cos(\theta)$ . For the pusher force  $\mathbf{f}_t^e$ , we apply  $\text{clip}(\mathbf{f}_t^e, -10, 10)/10$  component-wise. In practice, this means that we limit the magnitude of the force reading along each axis to 10 N. When providing the predicted uncertainty from the state estimator as an RL policy observation, we use the standard deviations, clipped to the range  $[0, 1]$ . Note that in these cases where the RL policy receives the predicted uncertainty, we keep the reward function unchanged.

During RL policy training, we use a learning rate scheduler based on the KL divergence of the policy, as in [18]. The scheduler has a target KL divergence of  $7 \cdot 10^{-3}$ , a minimum learning rate of  $1.5 \cdot 10^{-4}$  and a maximum learning rate of  $10^{-2}$ . For the policy function, we use a neural network architecture with the following layers and corresponding sizes: linear (128) + LSTM (256) + linear (128) + linear (22), with tanh nonlinearities. The output consists of 22 logits that define the categorical distributions over the  $x$  and  $y$  velocities. For the value function, we use the same neural network architecture but replace the final linear (22) layer with a linear (1) layer that outputs the state value prediction. During training, when episodes reach the maximum horizon, thereby terminating, we use the value function prediction corresponding to the final observation to bootstrap the final reward. Table 4 summarizes the remaining RL hyperparameters for PPO [30] training. Finally, Fig. 6 shows the training performance of the privileged policy  $\pi_{priv}(s_t)$  in the occlusion-free environment.

Hyperparameter	Value
Rollout Steps	100
Parallel Environments	4000
Mini-batch Size	25000
Epochs	5
Clip Range ( $\epsilon$ )	0.2
Discount Factor ( $\gamma$ )	0.993
GAE Parameter ( $\lambda$ )	0.95
Entropy Loss Coefficient	0.01
Value Loss Coefficient	1.0

Table 4: PPO hyperparameters.

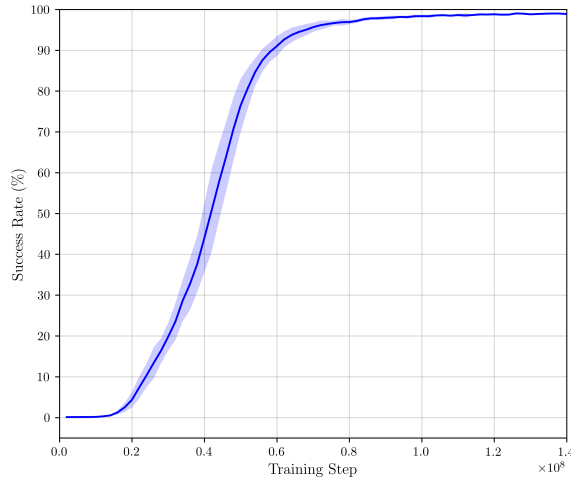


Figure 6: Training performance of the privileged policy  $\pi_{priv}(s_t)$ . We report mean and standard deviation across three random seeds.

## B.2 State Estimator

To train and evaluate the state estimators, we collect separate training, validation, and testing datasets containing  $7.5 \cdot 10^5$ ,  $1.5 \cdot 10^5$  and  $1.5 \cdot 10^5$  trajectories respectively. We process the trajectories scaling the pose and force measurements to the range  $[-1, 1]$  as discussed for the RL training in Appendix B.1. Additionally, Table 5 summarizes the training hyperparameters for the state estimator.

Fig. 7 and Fig. 8 show the validation loss when training the state estimator with the likelihood loss and the MSE loss, respectively.

Hyperparameter	Value
Mini-batch Size	30000
Epochs	110
Learning Rate	$10^{-3}$
Optimizer	Adam
Sequence Length	300

Table 5: State estimator hyperparameters.

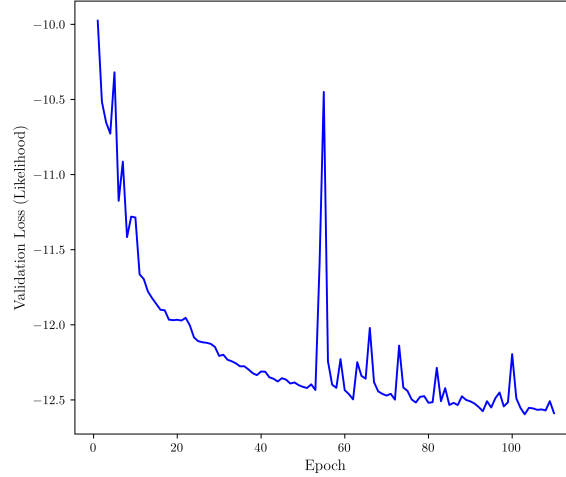


Figure 7: Estimator validation loss when training with the likelihood loss.

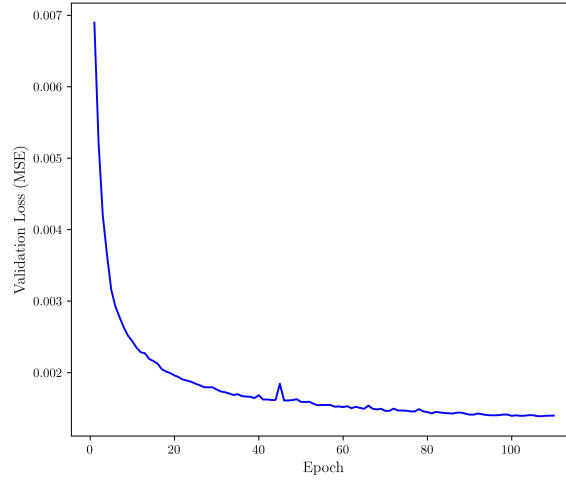


Figure 8: Estimator validation loss when training with the MSE loss.

### B.3 Behavior Cloning Baseline

To train the behavior cloning baseline discussed in Section 5.2, we collect new training and validation datasets using the last privileged policy checkpoint to provide optimal trajectories. We do not use a testing dataset since we evaluate the baseline directly in the planar pushing simulation environment. The training and validation datasets contain  $7.5 \cdot 10^5$  and  $1.5 \cdot 10^5$  trajectories respectively. We

470 process the trajectories to scale the pose and force measurements, add sensory noise, and introduce  
471 occlusions using the same procedure as for the state estimator.

472 The neural network architecture for the behavior cloning policy is the same as for the RL policy  
473 function, discussed in Appendix B.1. Hence, the output in both cases consists of 22 logits that  
474 define separate categorical distributions over the  $x$  and  $y$  velocities. We train the behavior cloning  
475 policy using a loss function defined as the sum of the cross-entropy between the predicted and target  
476 distributions for the  $x$  and  $y$  velocities. The training hyperparameters are the same as shown in  
477 Table 5. Finally, Fig. 9 shows the validation loss during training.

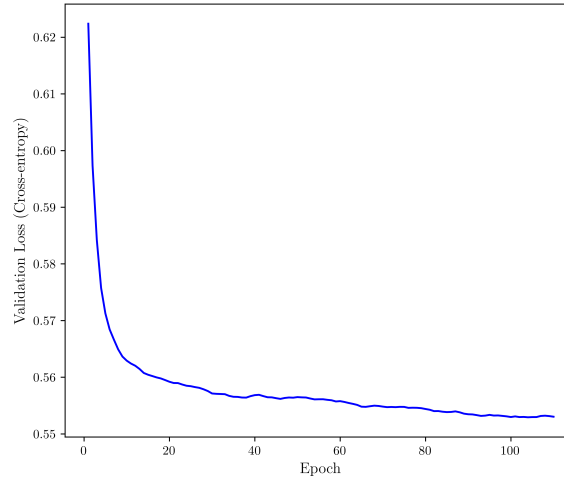


Figure 9: Behavior cloning policy validation loss.