

A Appendix

A.1 Limitations

1. **Camera calibration noise:** Shadow relies on accurate camera calibration in order to generate segmentation masks of the source and target robots. We investigate the robustness to camera calibration noise in Section A.4 and find Shadow deteriorates proportional to the scale of noise added. In future work, we could attempt to render Shadow more robust to camera calibration error via (modest) noise injection during training.
2. **Training is linear with number of embodiments:** We demonstrated that Shadow performs well on various target robots, but each individual policy we trained was for a specific source/target robot pair. Note, however, that training an additional policy for a new target does not require any additional data to be collected (i.e., data from a single source is sufficient for any number of targets). In future work, we also hope to evaluate Shadow’s ability to allow a single policy to generalize over more than two embodiments. Specifically, we could try overlaying segmentation masks of two or more target robot embodiments and evaluating if Shadow continues to perform well.
3. **Object occlusion:** In situations where the robot is (partially) occluded by objects in the scene, the overlaid segmentation mask will obfuscate these objects. As a result, the model may have difficulty disambiguating between cases where the robot is in front of an object versus behind it. In simulation, this can be easily addressed by using depth information from the original scene to only mask robot pixels behind occluding objects. In real, this can be addressed with real-time depth data (either by using a stereo camera or monocular depth estimation), but may be limited by the depth data quality.

A.2 Does Shadow perform well on the source robot?

The goal of Shadow is to develop a policy that performs well on the target robot despite never having collected data on the target robot. In this section, we investigate whether or not this same policy can also perform well on the source robot. In simulation, we run evaluations on the most challenging cross-embodiment scenario: different robot and different gripper. Results in Figure 9 demonstrate no significant change in performance between Shadow’s performance on the source and target robots across the majority of the tasks. For the Mug Cleanup task, we see increased performance on the source robot, which makes sense given that the target robot uses the Franka gripper which tends to jam against the drawer in the Mug Cleanup task whereas the source robot with the Robotiq gripper does not suffer from the same problem.

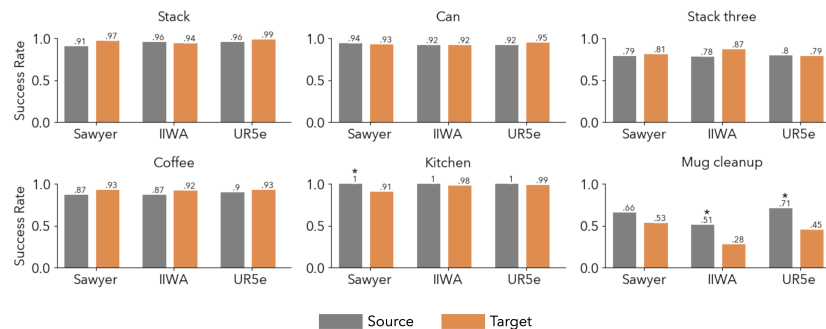


Figure 9: Shadow’s performance on six simulation tasks. The source robot is the Panda + Robotiq 2F85 gripper and the target robots use the Franka gripper. Shadow performs equally well on the source and target robots across the majority of tasks. (100 roll-outs per evaluation).

A.3 Does masking out robot pixels affect performance?

Shadow relies on masking out the robot’s pixels in the observation images. Our intuition for the feasibility of this approach stemmed from an experiment we ran in simulation that showed that replacing a source robot with its segmentation mask (i.e., discarding all RGB information about the source robot) had no effect on task performance (Table 2, left). In other words, the RGB values of the robot itself are not important for completing the tasks studied. Note that this section is independent of cross-embodiment learning - we simply replaced the source robot with its segmentation mask.

We ran a similar experiment on real robot hardware and came to the same conclusion: on all tasks, replacing the robot with its segmentation mask during training and evaluation did not affect task performance (Table 2, right).

	Simulation						Real world			
	Stack	Can	Stack 3	Coffee	Kitchen	Mug	Hex.	Cups	Stack	Mug
Vanilla	0.97	0.96	0.86	0.95	0.95	0.88	0.92	0.92	0.7	0.98
Masked	0.99	0.96	0.86	0.93	1.0	0.84	0.89	0.92	0.76	1.0

Table 2: Replacing the source robot with its segmentation mask does not affect task performance in either simulation or on real robot hardware. 100 roll-outs per evaluation in simulation and 25 roll-outs per evaluation in real. Vanilla: policy trained/evaluated on raw RGB images, Masked: policy trained/evaluated on images with the source robot replaced by its mask. No comparisons between Vanilla and Masked for a given task are statistically significant.

A.4 Sensitivity to camera calibration

Shadow relies on accurate camera calibration in order to generate segmentation masks of the source and target robots. We investigated the sensitivity of Shadow’s performance on real hardware to additional noise introduced to camera calibration extrinsics at evaluation time on the target robot (Table 3). As expected, the performance of Shadow deteriorates proportional to the scale of noise added.

Note that we did not deliberately add any camera calibration noise during training. In future work, we could attempt to render Shadow more robust to camera calibration error via (modest) noise injection during training.

Δx (m)	$\Delta \theta$	Mug	Hexagon
0	0°	0.98	0.76
0.01	5°	0.64	0.16
0.02	10°	0.20	0

Table 3: Sensitivity of Shadow to camera calibration error. We evaluated the success rate on real hardware on the target robot in different noise conditions (source: Franka robot with Robotiq gripper, target: UR5e robot with Robotiq gripper). Δx and $\Delta \theta$ denote the standard deviation of 0-mean Gaussian linear and angular noise deliberately added to the camera extrinsic parameters during evaluation, respectively.

A.5 Policy implementation details

Table 4 describes the hyperparameters used to train Diffusion Policy. The same hyperparameters were used for all methods, and we used the CNN-based Diffusion Policy architecture. In simulation, all tasks used the DDPM scheduler with 100 training diffusion steps and 100 inference steps. In the real world, all tasks used the DDIM scheduler with 100 training diffusion steps and 10 inference steps. All hyperparameters not listed are unchanged from the original Diffusion Policy paper.

	Task	To	Ta	ImgRes	Batch size	Lr	Epochs	Horizon
Real	Mug	2	8	240	256	1e-4	2000	200
	Cups	2	8	240	256	1e-4	2000	200
	Hexagon	2	8	84	256	1e-4	2000	200
	Blocks	2	8	84	256	1e-4	2000	200
Sim	Stack	2	8	84	256	1e-4	5000	200
	Can	2	8	84	256	1e-4	5000	200
	Stack Three	2	8	84	256	1e-4	5000	400
	Coffee	2	8	84	256	1e-4	5000	300
	Kitchen	2	8	84	256	1e-4	5000	750
	Mug Cleanup	2	8	240	256	1e-4	5000	400

Table 4: Hyperparameters used for diffusion policy. **To**: observation horizon, **Ta**: action horizon, **Epochs**: Number of training epochs, **Horizon**: max number of rollout steps

Our method can run at up to 20 Hz (for 84x84 images; or 10 Hz for 240x240), but we execute our policy on the real hardware at only 2Hz due to hardware limitations. In the set of experiments testing out a different gripper, we are forced to reduce our control frequency because the max control frequency of the Panda gripper is only 2Hz. In the set of experiments testing out a different robot, we prioritized hardware safety, as we were borrowing the UR5e, and chose to use the UR5e’s slow MoveL blocking command to send Cartesian end-effector pose commands because the MoveL command comes with inherent safety features. The MoveL blocking command forced us to slow our policy down to 2Hz.