Supplementary: Continuously Improving Mobile Manipulation with Autonomous Real-World RL

A. Videos

The main video summarizing our results can be found in result_video.mp4 in the zip folder. This depicts the robot performing each of the tasks we consider - moving the chair 1) with a table in the corner in the playpen, 2) with a table in the middle of the playpen, 3) picking up a dustpan and vertically orienting it such that it can stand up, 4) sweeping a paper bag into a target region. We also include timelapse videos which show how our approach adapts behavior over time.

B. Policy Training

For our experiments we run DrQ implemented in the official RLPD codebase open-sourced by Ball et al. [1]. Since we run image-based real robot experiments, we use learning algorithm hyperparameters (including for the image encoders) from Stachowicz et al. [4], which deployed RLPD for race car driving. The observations are first encoded into a latent space (separately for the actor and critic), and the processed latent is used by the critic ensemble or the actor. Details of the architecture for each of these, in addition to hyperparameters for training is provided in Table I.

We use both image and vector observations for learning. Each of these is processed by an image encoder or a 1-layer dense encoding for vector observations, and the corresponding latents are all concatenated together and then used as input for the actor or critic. Note that we use separate encoders for the critic and the critic. We use the architecture from Stachowicz et al. [4] for encoding each image source, without using any pre-trained embeddings, the network is retrained from scratch for each new experiment. There are 4 RGB image sources. The network encoders are provided with the last 3 frames for each image source, except for the goal image, since this remains fixed for the episode. The image sources are -

- Egocentric `front-left` image
- Egocentric `front-right` image
- Third-person fixed-cam current image
- Third-person fixed-cam goal image

We use (128,128) spatial resolution for the egocentric images, and (256,256) for the images from the third person camera. The latter uses a higher resolution since it is further away from the scene and objects appear smaller/less clear.

In addition, we have two vector observations -

- Body pose - We compute the (x,y,θ) position of the robot body in the SE(2) plane relative to the calibrated playpen frame (calibration details in section -F). The input to the network is 4 dimensional, consisting of $(x, y, \cos(\theta), \sin(\theta))$. We use $\sin, \cos$ transforms for the angle to avoid discontinuities in input, since $-\pi$ and π represent the same orientation.
- Hand pose - This contains the 6-dof end effector orientation of the hand relative to the base position.

TABLE I: Hyperparameters used in the experiments

Category	**Hyperparameter**	**Value**
Training	Batch size	256
	Update to Sample Ratio	4
Actor/Critic	Actor learning rate	3e-4
	Critic learning rate	3e-4
	Actor network architecture	2x256
	Critic network architecture	2x256
	Critic ensemble size	10
Image Encoder	Layer count	4
	Convolution size	3x3
	Stride	2
	Hidden channels	32
	Output latent dim	50

There are certain learning parameters that are tuned separately for each environment, which we list in Table II. This was mainly to balance the exploration-exploitation trade-off for learning new behavior, and pertain to the weight placed on entropy maximization in DrQ (temperature and target entropy), or to handle sparse rewards (number of min Q functions). We use a maximum episode length of 16 for the chair and sweeping tasks, and 8 for the dustpan task, since it has sparse reward.

TABLE II: Environment-tuned Hyperparameters

Env	#MinQ	Temp LR	Init Temp	Target Entropy
Chair	2	1e-4	0.5	-2
Dustpan	1	1e-3	0.1	-2
Sweeping	2	1e-4	0.1	-4

C. Rewards

1) Detection-Segmentation

For each task, there is an object of interest, the state of which is used to compute the reward. We specify the object using a text prompt, which is used by the detection model to obtain a bounding box. This is then used to condition the Segment Anything [2] model to obtain a 2D object mask. For text-based detection we use either Grounding-Dino [3] or Detic [5]. For Grounding-Dino, we append the task-specific prompt to the list of class names in COCO [**?**] (to avoid cases of false positive detection), and we use Detic with `objects365` vocabulary class names. The task-specific text prompts we use are 'chair' for the chair tasks, 'red broom' for the dustpan standup task, and 'box.bag.poster.signboard.envelope.tag.clipboard.street_sign' for the sweeping task. The object of interest in the sweeping task is a paper bag being swept and we use many different possible matching text descriptions since it is detected as different classes due to its deformable nature. We list the detection model and the confidence threshold for a detection to be accepted for each task in Table III.

TABLE III: Detection Settings

Env	Detection Model	Confidence Threshold
Chair	Grounding-Dino	0.4
Dustpan	Grounding-Dino	0.2
Sweeping	Detic	0.1

Once we obtain object masks, we can obtain the corresponding object point-cloud using depth observations. Some detections are rejected based on estimated position, eg: if there is a detection of an object outside the playpen. This filtering is essential since the robot often picks up on known infeasible objects, eg: the box in the middle of the playpen, or some chairs outside the railings.

2) *Reward Function*

Chair-moving tasks: For this task, we compute reward at every timestep of the episode. Given the estimated chair point cloud using the detection-segmentation system along with depth observations, we estimate the center of mass x_t and the yaw rotation w_t. Given the goal position g and orientation g_w (extracted from the goal image), we compute position x_{diff} and yaw difference w_{diff} norms. Then the reward is given by :

$$\begin{aligned} r_{\text{position}} &= -x_{\text{diff}} + e^{(-x_{\text{diff}})} + e^{(-10 \cdot x_{\text{diff}})} \\ r_{\text{ori}} &= e^{(-w_{\text{diff}})} + e^{(-10 \cdot w_{\text{diff}})} \\ \text{Total Reward} &= r_{\text{position}} + r_{\text{ori}} \end{aligned}$$

Dustpan Standup In this task, it is difficult to provide reward when the robot is interacting with the dustpan, since the detection model fails to pick up on the dustpan from the third person or egocentric image observations. We can measure reward at the end of the episode (when the robot has released its grasp) to detect the dustpan and estimate the center of the handle x_T, and provide a large bonus if the height of the handle (z component of x_T) is above a set threshold. To prioritize faster task completion, we use an alive penalty of -0.1. The robot can terminate the episode earlier by releasing its gripper and letting go of the handle.

$$\begin{aligned} r_{\text{penalty}} &= -0.1 \\ r_{\text{bonus}} &= 10 \text{ if } x_t \text{ height} \geq \text{thresh} \\ \text{Total Reward} &= \begin{cases} r_{\text{penalty}}, & \text{if timestep } t < T \\ r_{\text{bonus}}, & \text{if end of episode, timestep } T \end{cases} \end{aligned}$$

Sweeping: Similar to the chair task, we compute reward at every timestep of the episode. We estimate the point cloud of the paper bag, let its center of mass be denoted by x_t. The target region is a rectangle, denoted by G_r. Let $d(x, G_r)$ denote the distance from position x to the closest corresponding point on the rectangle given by G_r. Then the reward is given by:

$$\begin{aligned} r_{\text{distance}} &= -0.2 \cdot d(x_t, G_r) + e^{(-10 \cdot x_{\text{diff}})} \\ r_{\text{progress}} &= 10 \cdot \max(0, d(x_{t-1}, G_r) - d(x_t, G_r)) \\ r_{\text{bonus}} &= \begin{cases} 10, & \text{if } d(x_t, G_r) = 0 \\ 0, & \text{else} \end{cases} \\ \text{Total Reward} &= r_{\text{distance}} + r_{\text{progress}} + r_{\text{bonus}} \end{aligned}$$

3) *Success Criteria*

The results we show for continual improvement during training, as well as the evaluation of the final policies report success rate. Success is defined for an episode in the following manner for each of the tasks -

- Chair tasks - If the max reward obtained in the epsiode is above 1. This implies the chair is very close to its target.
- Dustpan Standup - If the episode ends with a reward of 10 (indicating the dustpan is standing up).
- Sweeping - If the episode ends with a reward of 10 (indicating the paper bag is swept into the goal region).

D. Priors

For the chair moving tasks we use RRT* for planning a path in SE(2) space with a simplified model that only has 2D occupancy of the top surface of the table, and is not aware of the chair, or robot-chair or chair-table interactions. This generates a set of way-points for the target position of the center of mass of the robot in SE(2) space, in global coordinates. We use coordinate transforms to convert these targets to be in the robot's body frame in order to use the same action space as the reactive RL policy. We are able to perform this computation since we know the robot's body position in global coordinates. Specifically, we have $W_{\text{body}} = W_{\text{global}} * T^{-1}$, where W_f denotes the way-point with respect to frame f and T is the matrix transform of the robot body center of mass with respect to the global coordinates.
For sweeping, the prior is simply to stay within 0.5m of the last detected location of the paper bag. For dustpan standup we use a simple procedural function to generate trajectories to create a prior dataset, which we detail in Algorithm 1

E. System Overview

We use a workstation with a single A5000 GPU to run RLPD online, which requires about 20GB GPU memory, mostly owing

Algorithm 1 Prior generation for Dustpan Pickup

```
Initialize Prior data buffer $\mathcal{D}$
Initialize Uniform noise distribution $\mathcal{U}$ with limits : $(-0.1, -0.1, -1) \rightarrow (0.1, 0.1, 1)$
for $N = 1$ to Number of episodes do
    Initialize action list $\mathcal{A} = []$
    Set yaw hand rotation $\omega$ to either +0.5 or -0.5
    for $t = 1$ to episode len do
        Set vertical hand action $z$ to be either +0.2 or -0.2
        Add $(z, \omega, 0) + (n \sim \mathcal{U})$ to $\mathcal{A}$
    end for
    Add $(-0.2, \omega, 0) + (n \sim \mathcal{U})$ to $\mathcal{A}$
    Execute $\mathcal{A}$ on the robot, record observations, add to $\mathcal{D}$
end for
return Prior data buffer $\mathcal{D}$
```

to all the image inputs that need to be processed. The detection and segmentation models are run on cloud compute on a single A100 GPU. The fixed third person camera images from the realsense are streamed to a local laptop. Communication between the laptop, workstation and cloud server is facilitated via GRPC servers, and the main program script is run on the workstation, which also controls the robot. Commands are issued to the robot over wifi using the SpotSDK provided by Boston Dynamics.

F. Map Calibration

We use the GraphNav functionality provided in the SpotSDK by Boston Dynamics for Spot robots for generating a map of the playpen. This involves walking the robot around with some fiducials (we use 5) in the arena. This needs to be performed only once, and is used to obtain a reference frame to localize the robot, which is useful to record body pose information and also to implement safety checks to make sure the robot is not executing actions that collide with the playpen railings. While Spot has inbuilt collision avoidance we implement an additional safety layer based on the generated map to clip unsafe actions that would move the robot too close to the playpen railings.

REFERENCES

[1] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient Online Reinforcement Learning with Offline Data. In *ICML*, 2023. 1

[2] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment Anything. *arXiv preprint arXiv:2304.02643*, 2023. 2

[3] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection. *arXiv preprint arXiv:2303.05499*, 2023. 2

[4] Kyle Stachowicz, Dhruv Shah, Arjun Bhorkar, Ilya Kostrikov, and Sergey Levine. FastRLAP: A System for Learning High-Speed Driving via Deep RL and Autonomous Practicing. *arXiv preprint arXiv:2304.09831*, 2023. 1

[5] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Phillip Krähenbühl, and Ishan Misra. Detecting Twenty-Thousand Classes Using Image-Level Supervision. In *ECCV*, 2022. 2