Continuously Improving Mobile Manipulation with Autonomous Real-World RL

Author Names Omitted for Anonymous Review. Paper-ID [451]

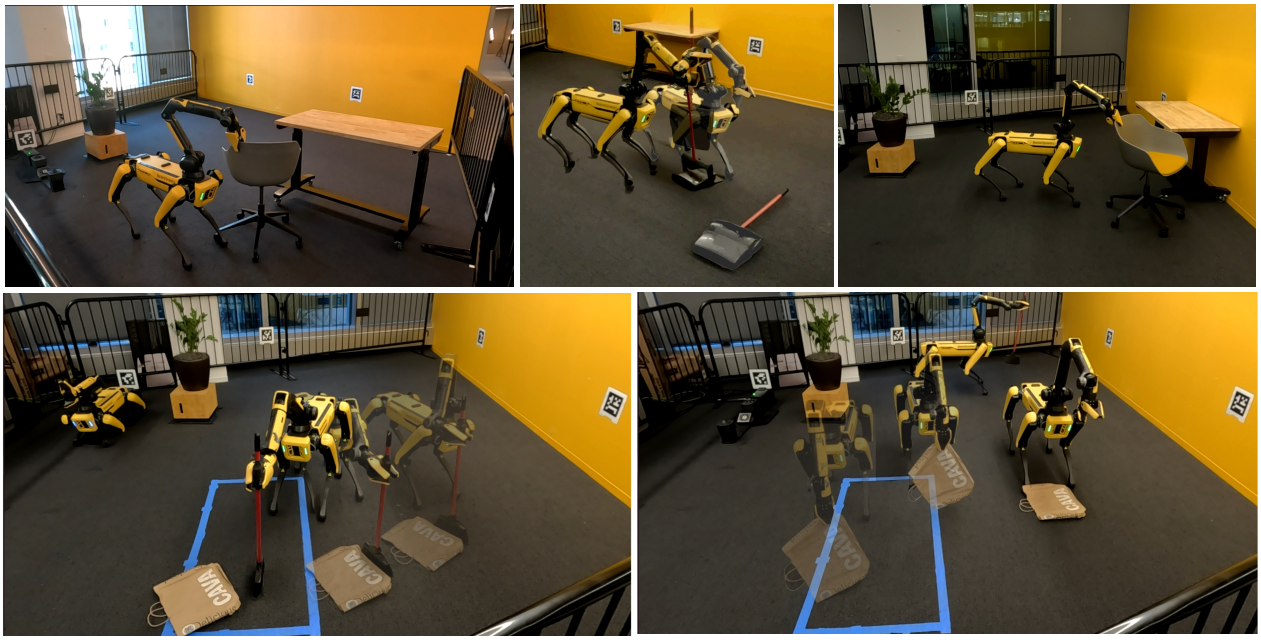

Fig. 1: **Continual Autonomous Learning**: We enable a legged mobile manipulator to learn a variety of tasks such as moving chairs (top, left and right), righting a dustpan (top, middle), and sweeping (bottom) via practice in the real world with minimal human intervention.

***Abstract*—To build generalist robots capable of executing a wide array of tasks across diverse environments, robots must be endowed with the ability to engage directly with the real world to acquire and refine skills without extensive instrumentation or human supervision. This work presents a fully autonomous real-world reinforcement learning framework for mobile manipulation that can both independently gather data and refine policies through accumulated experience in the real world. It has several key components: 1) automated data collection strategies by guiding the robot's exploration toward object interactions, 2) using goal cycles for world RL such that the robot changes goals once it has made sufficient progress, where the different goals serve as resets for one another, 3) efficient control by leveraging basic task knowledge present in behavior priors in conjunction with policy learning and 4) formulating generic rewards that combine human-interpretable semantic information with low-level, fine-grained state information. We demonstrate our approach on Boston Dynamics Spot robots in continually improving performance on a set of four challenging mobile manipulation tasks and show that this enables competent policy learning, obtaining an average success rate of 80% across tasks, a 3-4× improvement over existing approaches.**

I. Introduction

As robots transition from the structured confines of fully mapped industrial settings into the dynamic and unstructured realm of our daily lives, there is an increasing need to build generalist systems capable of executing a wide array of tasks across diverse environments. While visuomotor policies trained with reinforcement learning (RL) have demonstrated significant potential to bring robots into open-world environments[24, 25, 30], in practice, they first require training in simulation [2, 9, 10, 20, 49, 57]. However, it is challenging and not scalable to build simulations that capture the unbounded diversity of real-life tasks, especially involving complex manipulation. What if we instead adopt a strategy where learning occurs through direct engagement with the real world, without extensive environmental instrumentation or human supervision during the training process? This would allow robots to acquire and refine skills for various tasks continuously. In this work, we present a fully autonomous real-world reinforcement learning framework for mobile manipulation that can both independently gather data and refine policies through accumulated experience. We address multiple challenges for building an effective real-world learning system.

Challenge 1: Automated collection of useful data: Consider a complex, high-dimensional system like a legged mobile manipulator operating in open spaces where undirected actions often do not affect any meaningful change in the environment. The first challenge in building an effective real-world learning system is in autonomous, task-relevant data collection because good robot autonomy does not imply the resulting data has a

useful learning signal. For example, we would like to avoid the robot simply waving its arm in the air without interacting with objects if its goal is to acquire manipulation skills. While such a system could, in theory, learn sophisticated mobile manipulation strategies given enough data, we propose using off-the-shelf visual models to design automated strategies that make learning in the real world feasible by guiding the robot's exploration toward object interactions. In particular, we design automatic grasp and navigation procedures (described as `auto-grasp` and `auto-nav` in Algorithm 2), which utilize detection systems and pre-existing skills to move the robot near and grasp task-relevant objects. But this is not sufficient because the legged manipulator does not know what to do after grasping, and in many cases, even grasps need to be applicable to the particular task, e.g., holding a dustpan from its handle to pick it up (see Figure 1).

Challenge 2: How to ensure diverse practice? The second challenge is how to allow the robot to purposely practice achieving goals from diverse initial states without human resetting. That is, once the robot is close to its goal, it does not get to practice the task from states that are further from the goal. For instance, consider a robot tasked to move furniture. The robot may learn to move a piece of furniture to its target location; however, now that the furniture is very close to the goal, continuing to practice the task from this starting state will not yield further benefits. Instead, if the environment state could be reset back to the initial state distribution, the robot could practice repeating its success. In the absence of such resets, how can we enable autonomous robots to return to the harder initial state distribution for practicing tasks? The approach we use is to set up '*goal-cycles*' [16, 17, 21], where we switch the goal once the robot has made sufficient progress on the previous one, or spent a budget of a fixed interval of trajectories attempting it. Hence, the goals serve as resets for one another, and this multi-goal learning setup ensures that the robot does not stagnate in a limited region of the state space near any particular goal. One difficulty of this setup is that learning potentially very different skills to achieve these multiple goals increases the burden on the learning approach and may slow down execution as the robot cannot practice only one skill full-time. So, we devise a multi-robot setup (i.e., two Spot robots with arms) to mitigate this challenge, e.g., while one robot focuses on the task of sweeping, the other one picks up the swept object and drops it further away.

Challenge 3: Efficient control in the real world: Even with a favorable initial state distribution, policy learning poses a daunting challenge due to large observation and action spaces. This challenge is especially severe in the case of legged mobile manipulation, where the robot needs to move and simultaneously maintain contact with objects and retain control. Our approach expedites learning control policies by leveraging basic task knowledge present in behavior priors. These priors can take the form of planners with a simplified incomplete model or automated procedurally generated behaviors. It is important to note that while these priors bootstrap learning and help provide a signal for learning, particularly in the early stages, the priors might not be very competent at performing the task, owing to their simplicity. For example, an RRT* planner [29] for moving in the x-y plane does not have a model of robot-chair or chair-table interaction dynamics and will not be able to recover effectively from collisions. Similarly, such a planner will struggle trying to sweep a paper bag which requires maintaining contact with the bag, adapting to different ways the bag might move, and still moving the robot towards the goal. In our experiments, the average success rate of the prior is just 20% across tasks but as low as 5% for the challenging task of sweeping. In contrast, our learning-based approach enables an average success rate of 80%, a 4× improvement. Hence, the priors are not a substitute for learning controllers but rather serve to expedite learning by structuring exploration.
Challenge 4: Defining rewards in the real world: For the system to benefit from the previously described structure and get better at performing tasks, it must evaluate the relative benefit of different actions by receiving reward feedback from the environment. Providing reward supervision in the real world often requires physical instrumentation in the form of specialized sensors [39, 56] or needs humans in the loop [11, 32, 42]. Furthermore, the ability of these robots to keep collecting data and learning to improve is bottlenecked by how expensive or difficult it is to scale these approaches. There has been some work on completely self-supervised learning systems with some extensions to robotics [35, 37], but these approaches are challenging to deploy on complex tasks due to intractability, underspecification, and misalignment. In this work, we seek a *flexible* way for humans to specify objectives for *arbitrary* tasks. To this end, we devise a generic reward modeling recipe that combines human-interpretable, semantic information, i.e., text-based detection and segmentation models, along with low-level, fine-grained state information, i.e., vision and depth-based observations for object estimation. Despite yielding noisy estimates, we find the resulting reward is sufficient to allow the robot to learn challenging tasks.

The main contribution of this work is a general approach for continuously learning mobile manipulation skills directly in the real world with autonomous RL. The main components of our approach involve: (1) task-relevant autonomy for collecting data with useful learning signals, (2) efficient control by integrating priors with learning policies, and (3) flexible reward specification combining high-level visual-text semantics with low-level depth observations. Our approach enables a Boston Dynamics Spot robot to continually improve in performance on a set of 4 challenging mobile manipulation tasks, including moving a chair to a goal with the table in the corner or center of the playpen, picking up and vertically balancing a long-handled dustpan, and sweeping a paper bag to a target region. Our experiments show that our approach gets an average evaluation success rate of about 80% across tasks, which is a **4× improvement** over using either RL or the planner individually. Notably, on the challenging sweeping task, our method is the only one to learn a competent policy compared to using just a prior, offline RL on the prior data or RL without priors.

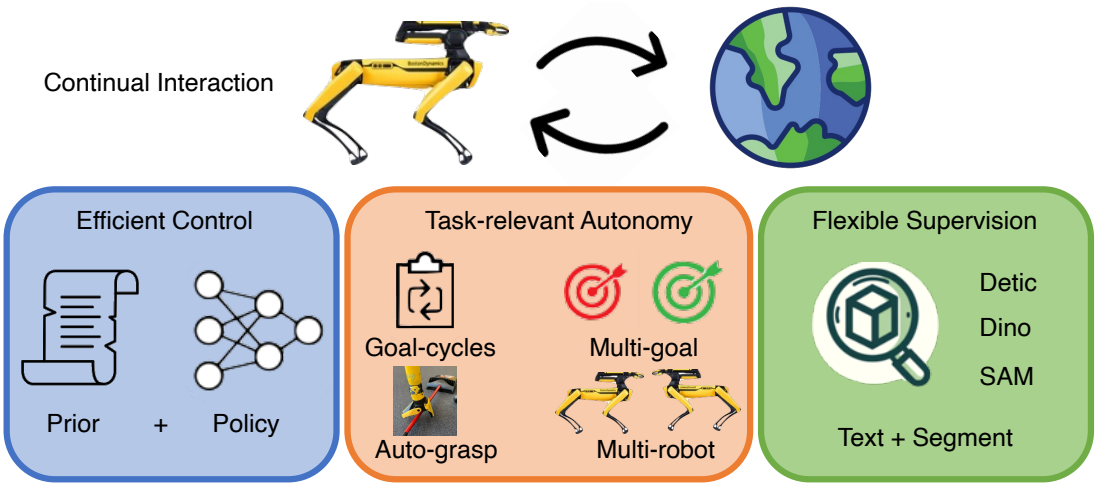

Fig. 2: **Method Overview**: The main components of our approach for robots to continually practice tasks in the real world. **Left**: Efficient control by aiding policy learning with basic task knowledge present in behavior priors in the form of planners with a simplified model or automated behaviors. **Center**: Task-relevant autonomy to ensure collection of useful data via `auto-grasp`, and maintaining state diversity via automated resets using multi-goal and multi-robot setups. **Right**: Flexible reward supervision that combines human-interpretable semantic detection-segmentation information with low-level, fine-grained depth observation.

II. Related Work

A. Autonomous Real-World RL

A large number of works have focused on directly training robots with RL in the real world [24, 25, 30, 36]. To enable autonomous systems, prior works have studied reset-free RL with forward-backward policies [41], and a graph structure of sub-tasks that serve as resets for one another [16, 55]. This has been studied in table-top settings involving arms and dexterous hands, and in our work, we aim to adapt these ideas for mobile manipulation problems. There has been work on real-world RL training for legged locomotion [18, 44, 52], where steps have been taken to address autonomy, such as training reliable reset policies [43]. Approaches have also aimed to tackle autonomous mobile manipulation [47, 54], but are mostly constrained to single tasks. Moreover, task specification is a big challenge [62] if one wants to enable autonomy. A natural way to specify tasks is to leverage language goals and combine these with large-scale visual models [26], conditioned on open-vocabulary prediction [5, 31, 60]. However, these approaches require some level of prompting and guidance from a human.

B. Mobile Manipulation

Mobile manipulation methods balance many trade-offs to enable complex reasoning and control over both manipulation and navigation. In the 2015 DARPA Robotics Challenge Finals, mobile manipulation solutions primarily relied on pre-built object models and task-specific engineering to enable mobile manipulation [28]. More recent work modularizing tasks into skill primitives and interacting with those primitives using flexible planners, including large language models, has enabled more generalization outside of pre-coded tasks [3, 47, 50, 51]. Imitation learning approaches to mobile manipulation enable joint reasoning over manipulation and navigation actions and generalize across broad sets of tasks [1, 7, 13, 15, 40]. However, imitation learning requires an expensive collection of expert trajectories. In contrast, RL methods can learn from experience and have also enabled whole-body control of mobile manipulators [12, 23, 57]. Decomposing the action space over which the RL policy operates enables more tractable and efficient learning of long-horizon mobile manipulation skills [14, 34, 53, 59]. In our work, we move beyond tasks that involve picking and placing to instead learn skills that require more coordination between the legs and arms, e.g., moving chairs or sweeping. Doing so in simulation is challenging due to the difficulty in modeling robot-object dynamics. Hence we train our system directly in the real world. Learning is made feasible and efficient using task-relevant autonomy and by incorporating behavior priors.

III. Continuously Improving Mobile Manipulation via Autonomous Real-World RL

We design our approach to allow robots to autonomously practice and efficiently learn new skills with minimal human intervention. For each task, we assume that there is a particular object of interest that the robot must learn to manipulate. To do so, it must first interact with the object, so we enable this at the beginning of every episode using an `auto-grasp` procedure (or just `auto-nav` since some tasks do not require grasping). We then execute trajectories using a combination of a prior and the control policy to collect data for learning. Rewards are obtained for these samples by estimating object position, using a combination of text-prompted detection, visual segmentation, and depth observations. If the object state is very close to its intended goal, we switch to trying to reach a different goal, serving as a 'reset' for the next task. Further, we also switch goals after a fixed number of episodes to prevent stagnation of the robot in any part of the state space and to ensure it can keep learning effectively. The overview of this entire process

Algorithm 1 Autonomous RL for Legged Mobile Manipulation

Require: Third person image I_C, egocentric images $\{I_{EC}\}$
Require: Detection-segmentation models $M(.)$
Require: Behavior prior $P(.)$
Initialize Data buffer $\mathcal{D}$, RL policy π_θ
Initialize task goal $\mathcal{G}_\mathcal{T}$ with goal object state $g_\mathcal{T}$
Initialize trajectories per task K, episode horizon H
while training **do**
  for trajectory 1:K **do**
    Approach object using Autograsp or AutoNav (Alg. 2)
    for timestep 1:H **do**
      Use policy $\pi_\theta(.)$ and prior $P(.)$ for either separate (Eq. 1), sequential (Eq. 2) or residual (Eq. 3) control.
      Compute reward r_t using $M(I_C, \{I_{EC}\})$
      Add $(o_t, a_t, o_{t+1}, r_t) \mapsto \mathcal{D}$
      Sample batch $\beta \sim \mathcal{D}$ to update policy π via RL
    end for
    (optional) If distance$(x, g_\mathcal{T}) \leq \epsilon$, break
  end for
  Switch task goal $\mathcal{G}_\mathcal{T}$

end while

Algorithm 2 Autograsp/AutoNav

Require: Third person image I_C, egocentric images $\{I_{EC}\}$
Require: Detection-segmentation models $M(.)$
Require: Navigation skill $N(.)$, Grasping skill $G(.)$
Use $M(I_C)$ to attempt object state (x) estimation
while object not detected **do**
  Use $M(\{I_{EC}\}))$ to attempt x estimation
  Step robot base by $(x, y, \theta) \sim$ Unif.$[-1, 1]$
end while
Use $N(.)$ to navigate close to x
Use $G(.)$ to grasp the object (optional)

is summarized in Alg. 1. Next, we describe further details of different components of our approach, involving task-relevant autonomy, efficient control by integrating priors with learning policies, and flexible reward specification.

A. Task-Relevant Autonomy

Auto-Grasp/Auto-Nav: We outline the process for this procedure in Alg. 2. Every episode begins with the system attempting to estimate, move to, and/or grasp the object of interest for the task. Given an image, object estimation consists of first running open-vocabulary text-based detection (e.g. Detic [61] or Grounding DINO [33]) to obtain bounding boxes, which are then used to prompt a vision segmentation model, like Segment Anything (SAM [26]). The resulting object mask can be used along with depth observation from calibrated cameras in the environment (including the robot's egocentric vision sensors) to obtain the 3-D pointcloud of the object (see Fig. 5). Detection is first run on the image from the fixed camera, and if the object is not detected, we look at images from the egocentric cameras and move the robot by a random step $(x, y, \theta) \sim$ Unif.[-1,1] in the SE(2) plane until the object is found.

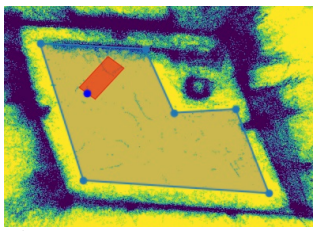

Fig. 4: Collision map of the playpen used by the SE(2) navigation planner. The table is added to this map when included in experiments.

Once the object position has been detected, we can use pre-existing navigation and grasping skills to move the robot close to the object. For navigation we use RRT* to plan in SE(2) space given the obstacles, using the collision map of the playpen as shown in Fig. 4. The red region denotes the estimate of the robot's position in the x-y plane, with the blue marking denoting its heading. The map is generated before experiments as part of calibration, by walking the robot around different parts of the playpen.

For grasping, we use the grasp API from the Boston Dynamics Spot SDK. This grasp is generated via a geometric algorithm that fits a grasp location with a geometric model of the gripper, scores different possible grasps and picks the best one. We do not constrain the type of grasp, or on which portion of the object the grasp is performed. We found that this allows the robot to keep practicing regardless of which position or orientation the object might end up in as a result of continual interaction.

Goal-Cycles: To ensure that autonomous robots can keep learning effectively after initial success, we set up 'goal-cycles' within tasks. We show the different goal states used in each of the 4 tasks we consider in Fig.3. For certain tasks like Long Handled Dustpan PickUp (Fig.3 e,f), we do not need an explicit goal cycle, since the handle can be picked up both from the ground and when it is standing, and the robot can continue to practice the task from roughly the same starting state distribution. In the case of the chair moving tasks (Fig.3: a-d), the robot alternates between goals that are far apart in the x-y plane. For the sweeping task (Fig.3: g-h), we use a multi-robot setup for the goal cycle, where one robot holds the broom and needs to sweep the paper bag into the target region (denoted by the blue box), while the other needs to pick up the bag and drop it back into the region where it can be swept. Since we only need learning for the sweeping skill, the robot that picks up the bag runs the `auto-grasp` procedure.

B. Prior-guided Policy Learning

Incorporating Priors: We enable efficient learning by leveraging priors that utilize basic knowledge of the task. This removes the burden from the learning algorithm from having to rediscover this knowledge, and instead focus on learning other additional behavior needed for solving the task. The priors can take the form of a planner or automated, procedurally generated behavior. For example, an RRT* planner with a simplified 2D model can help an agent move between two points in the x-y plane while avoiding obstacles. Starting with this prior, using RL can help the robot learn to recover from

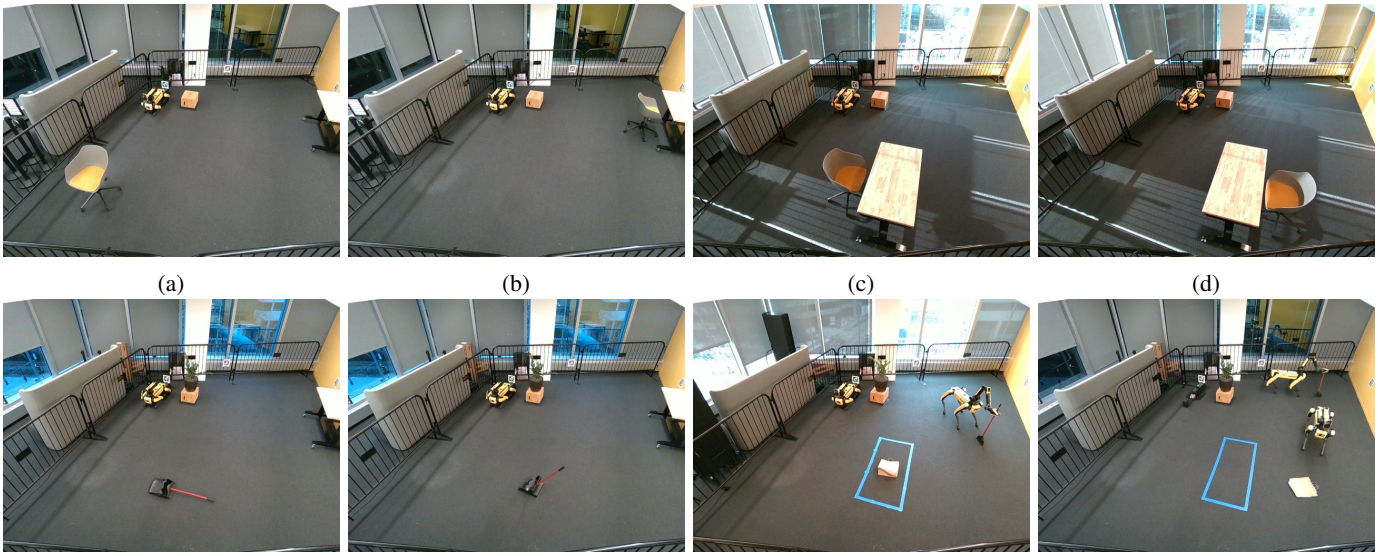

Fig. 3: **Task Goals**: Visualization of the states that form goal-cycles for each of the 4 tasks we consider - (a-b): Chair Moving with a table in the corner, (c-d): Chair Moving with a table in the middle, (e-f): Long Handled Dustpan Pickup, (g-h): Sweeping

collisions and deal with dynamic constraints not represented in the model. Concretely, the prior is a function $P(.)$ that takes in an observation o_t and produces an action a_t, similar to a policy $\pi(a_t|o_t)$. We have three ways of incorporating this - the prior and policy can be executed separately, where we alternate between the two, we can execute them sequentially, where the policy is executed after the planner or we can adopt residual control of the policy around a planner constraint.

1. *Separate*: Trajectories are collected independently using either the prior or the policy.

$$\begin{aligned} \text{Prior:} \quad & \{P(a_0|o_0), \ldots, P(a_T|o_T)\} \\ \text{Policy:} \quad & \{\pi(a_0|o_0), \ldots, \pi(a_T|o_T)\} \end{aligned} \tag{1}$$

Instead of learning entirely from scratch, we incorporate the (potentially) suboptimal data from the prior into the robot's data buffer to bootstrap learning. Intuitively, the prior is likely to see a higher reward than a completely randomly initialized policy, especially for sparse reward tasks. In fact, previous work has found incorporating data from expert demonstrations [38] or even from other policies that are suboptimal with respect to a new task [4, 22, 45, 46, 48] to vastly decrease the amount of online data required to learn. We adopt the latter strategy, making no assumptions on the optimality of the prior, and bootstrap learning via incorporating its *data*. In practice, we first collect trajectories using the prior and use it to initialize the data buffer used for training the online RL policy $\pi(.)$.

2. *Sequential*: In addition to providing data with better signals to the learning process, priors can reliably make reasonable progress on a task. This is because they often generalize well, for example, an SE(2) planner will make reasonable progress in moving a robot between any two points in the x-y plane, even when it performs constrained manipulation. We would need to sample many times from the prior to distill this information purely via the data buffer. Hence a more direct approach is to utilize the prior along with the policy for control. We do this by sequentially executing the prior, followed by the policy. That is, trajectories collected in this manner take the form:

$$\{P(a_0|o_0), .., P(a_L|o_L), \pi(a_{L+1}|o_{L+1}), .., \pi(a_T|o_T).\} \tag{2}$$

This effectively uses the prior to structure the initial state distribution of the policy, which makes learning easier. The data collected by the prior is also added to the data buffer, allowing the policy to learn from these transitions.

3. *Residual*: In certain cases, the prior might not be robust enough to deploy directly but nonetheless provide reasonable bounds on what actions should be executed. For example, for sweeping an object, the robot's base should roughly be in the vicinity of the trash being swept, but this does not prescribe what exact actions to take. An approach to utilizing such a prior is to use residual control, where a policy adjusts the actions of the prior at every time step before being executed. These trajectories take the form:

$$\{P(a_0|o_0) + \pi(a_0|o_0), \ldots, P(a_T|o_T) + \pi(a_T|o_T)\} \tag{3}$$

RL Policy Training: The RL objective is learn parameters θ of a policy π_θ to maximize the expected discounted sum of rewards $R(s_t, a_t)$:

$$J(\pi_\theta) = \mathbb{E}_{\substack{s_0 \sim p_0 \\ a_t \sim \pi_\theta(a_t|s_t) \\ s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)}} \left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t), \right] \tag{4}$$

where p_0 is the initial state distribution, $\mathcal{P}$ is the transition function and γ is the discount factor. For sample efficient learning that effectively incorporates prior data, we use the state-of-the-art model-free RL algorithm RLPD [4]. RLPD is an

off-policy method based on Soft-Actor Critic (SAC) [19], which samples from a mixture of data sources for online learning. Since our observations consist of raw images, we incorporate the image augmentations added by DrQ [58] to the base RL algorithm. Like REDQ [8], RLPD uses a large ensemble of critics and in-target minimization over a random subset of the ensemble to mitigate over-estimation common in TD-Learning (we refer the reader to the RLPD paper for further explanation). In our experiments, we use an ensemble of 10 critics and an update-to-data ratio of 4. For dense reward tasks, we minimize over two critics, while for sparse reward tasks, we simply sample one critic for the target value. The observation space for our policy training consists of three image sources: the fixed, third-person camera, the egocentric front-left and front-right cameras on the quadruped, and the body position, hand position, and goal image. The policy sees the three most recent images for each image source at each timestep. We use the JAX implementation [6] of this algorithm open-sourced by Ball et al. [4], which allows for fast training.

C. Flexible Supervision via Text-Prompted Segmentation

Fig. 5: **Grounded SAM/Detic Visusalization**: We run open-vocabulary text-based detection to obtain bounding boxes, followed by a segmentation model to obtain an object mask, visualized for chair (left) and sweeping (right). This can be used along with depth observation from calibrated cameras in the environment (including the robot's egocentric vision sensors) to obtain 3D object point cloud.

Each task has a corresponding object of interest that needs to be manipulated into a target configuration. Given the text label of this object, we can use detection models like Detic [61] to obtain the corresponding bounding box. This can then be used to condition segmentation models like Segment-Anything [26] to obtain the corresponding object mask. Using depth observations and the calibrated camera system for either the egocentric or fixed third-person cameras, we can then obtain the point cloud of the object. This can then be used to obtain an estimate of the object's state, and compared to the desired goal state. We provide more detail on the form of the rewards used in Section IV and provide full details on the exact prompts, detection and segmentation models, and reward functions for each task in the supplemental materials. We find that even if this estimation is noisy, it is good enough to enable us to learn effective control policies.

IV. Experimental Setup

For our experiments, we run continual autonomous RL in a playpen of about 6×5 meters using the Boston Dynamics Spot robot with an arm and simple gripper for manipulation. The playpen is enclosed with metal railings for safety since the robots inside operate continuously and autonomously. We use an IntelRealSense D455 mounted above the playpen for third-person fixed RGBD camera observations. The Spot has 5 egocentric body cameras that detect images along with depth from the front-left, front-right, left, right, and back, as well as a hand-camera located inside its gripper. We use all the body camera images when searching for objects as part of `auto-nav`, and use the hand-camera for grasping. Only the front-left and front-right images are used when training control policies. There is a docking station for the robot within the playpen, and it has the ability to autonomously auto-dock when it runs low on charge, and continue practicing the task once its battery has recharged.

We set up four different mobile manipulation tasks that move beyond simple picking and placing of small objects, requiring control of both the body as well as the arm: chair moving, with a table either in the 1) corner or the 2) center of the playpen, 3) vertically balancing a long-handled dustpan, and 4) sweeping. Mobility of the robot system is also essential for automated resets critical for continual practice. The states that depict the endpoints of the goal cycles are shown in Fig. 3.

Chair Moving - Corner Table: This task requires the robot to grasp a chair, and move it to a goal location. We first consider moving the chair to align with a table placed against the wall and the corresponding reset goal state on the other side of the playpen (Fig.3 a-b). The action space for this task is 5 dimensional, with base (x, y, θ) control and (x, y) control for the hand relative to the base. The prior used in the task is the RRT* planner in the x-y plane, with no knowledge of the chair. This planner is useful for navigating from one point to another, but it does not consider object dynamics or interaction. The prior is executed sequentially, i.e., we first run the planner, and then the RL policy for every episode, and the data it samples is added to the data buffer used for RL training. The reward takes into account the position and yaw distance between the current chair position and the target chair position at each timestep.

Chair Moving - Middle Table: We next consider a more challenging variant of the above task, where the robot needs to maneuver around the table placed in the middle of the playpen, with goal states as shown in Fig.3 c-d.The increased difficulty is because collisions between the chair and table base are much more frequent, and also because the robot has to operate in a much tighter space. The action space, prior used, and reward are the same as in the previous task, except that the RRT* planner has the 2D occupancy of the table to avoid collisions.

Long-handled dustpan Pickup: The task is to lift up the long handle of a dust-pan, and then to vertically balance it so that it can stay upright on its base. The action space is 3 dimensional, consisting of $(z, \text{yaw}, \text{gripper})$ commands for the hand. The gripper open action is used to terminate the episode earlier since the system resets if the grasp is lost. This task has sparse rewards since the handle is not detectable by segmentation models when the robot is interacting with it, so rewards cannot be computed during the episode. Instead,

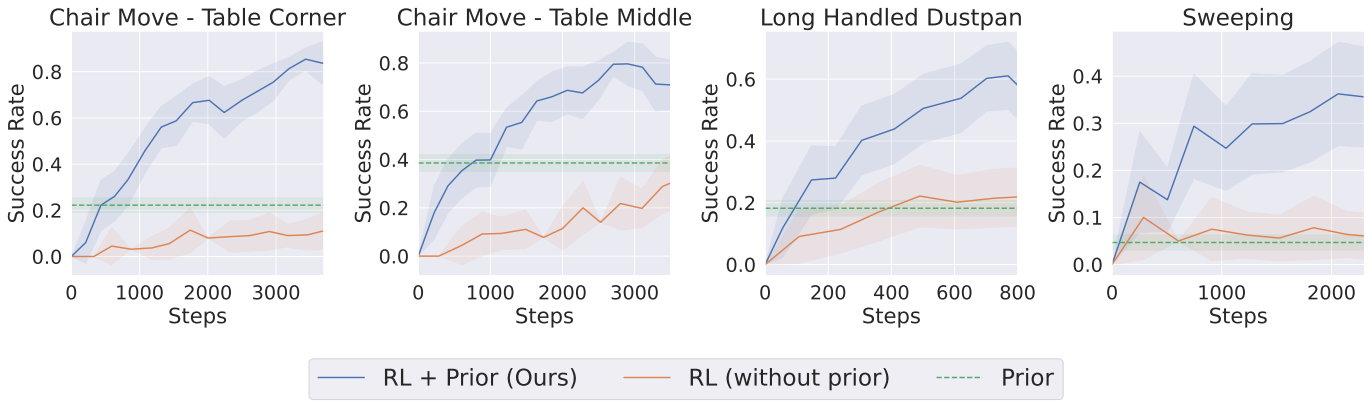

Fig. 6: **Continual training improvement**: Success rate vs number of samples for our approach and RL without priors, and average performance of the planner across all samples. We see that our approach continuously improves with experience across tasks, learning much faster than RL without priors, and attaining significantly higher performance than just using the prior.

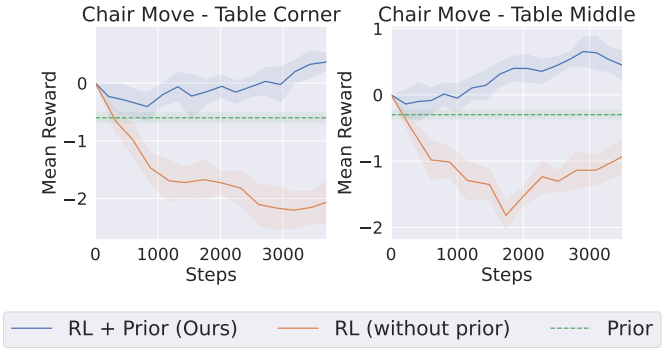

Fig. 7: **Training mean reward**: Mean reward vs number of samples for the chair moving tasks. The negative average reward for RL without priors indicates that the robot is often far from the goal location.

we detect the object at the end of the episode and then use its estimated height to provide a reward if it is upright. We include a small penalty at every time step to encourage learning faster solutions, and a large bonus if the handle is detected to be upright at the end of the episode. The prior used for this task is an automated procedure that tries to place the dustpan vertically. This is used independently from the policy; we first collect a large number of rollouts from the prior and use this to initialize the data buffer for RL training.

Sweeping: This task involves two robots, where one of the robots holds a broom in its gripper and needs to use it to sweep a paper bag into a goal region. The robot does not maintain firm contact with the paper bag but rather interacts with it using the broom. The other robot is tasked with resetting the paper bag by picking it up and dropping it close to the initial position for sweeping. Note that we only use learning for the robot that performs sweeping, while the other robot uses off-the-shelf visual models and the `auto-grasp` procedure to reset the bag. The action space for this task is 5 dimensional similar to the chair tasks, with base (x, y, θ) control and (x, y) control for the hand relative to the base. The prior involves detecting the paper bag's position and then staying within some distance of it while trying actions. Hence, this prior just biases the robot to stay close to the paper bag instead of moving to a different region of the playpen. The policy is executed in a residual manner around this prior constraint. For training the sweeping policy, we design a reward that takes into account the current distance of the paper bag to the goal region, the progress made since the last timestep (i.e., the improvement in how close the bag has moved to the goal), and a large bonus reward for when the paper bag ends up inside the goal region.

	Ours	Only RL	Only Prior	Offline RL
Chair-tablecorner	**1**	0.2	0.22	0.1
Chair-tablemiddle	**0.8**	0.5	0.38	0.2
Long-handle dustpan	**0.6**	0.2	0.18	**0.6**
Sweeping	**0.8**	0	0.05	0.1

TABLE I: **Evaluation Comparison**: The success rate of the final policy evaluated on different tasks. For evaluation, we use the deterministic policy instead of sampling from the stochastic distribution like in training. Our approach gets an average success rate of 80%, about 4X improvement over using only the prior or only RL.

In our experiments, we use the elements from task-relevant autonomy (i.e., `auto-grasp` and goal-cycles via multi-goal or multi-robot setups) for all methods, since this is essential to collect data and learn. Our investigation is primarily focused on evaluating if our combination of RL and the prior is effective. In addition to comparing to using only RL or only the prior, we also run offline RL on the data collected by the prior, specifically using IQL[27]. For each task, we specify success criteria corresponding to task completion. This corresponds to the chairs being moved to the goal locations shown in Fig.3:a-d, standing up the dustpan (Fig.3-f), and sweeping the paper bag into the goal region (Fig.3-g). Performance for the chair tasks is averaged across both location goals in each setting. We use the same network architectures for image processing, critic functions, policy, etc., for our approach, RL from scratch, and offline RL. Please see supplementary materials for further details, including the full reward functions, success criteria, procedural functions for priors, hyper-parameters for learning, and network details.


Fig. 8: **Left**: The prior (RRT* with incomplete model) gets stuck in a collision with the table and is unable to recover as the planner does not have a model of chair-table interaction dynamics. **Right**: Our approach effectively recovers from collisions to complete the task.

V. Results

Our real-world experiments test whether autonomous real-world RL can enable robots to continuously improve mobile manipulation skills for performing various tasks. Specifically, we seek to answer the following questions:

1) Can a real robot learn to perform tasks that require both manipulation and mobility in an efficient manner?
2) Does performance continually improve as the robot collects more data?
3) How does the approach of structured exploration using priors along with RL, compare to solely using the prior, or RL without constraints?
4) How does the policy learned via autonomous training perform when evaluated in test settings?

A. Continual Autonomous Improvement

From Fig. 6, we see that our approach of utilizing the combination of the prior and RL learns to continually improve with collected experience across all tasks. Note that in every instance, our approach starts out performing worse than the prior, and at the same level as RL without the prior. However, our approach learns significantly faster than RL from scratch by leveraging the prior. For some tasks, RL from scratch does improve in performance (e.g., chair move - table middle), but the rate of improvement is much slower than that of our method. We observe that in the chair moving experiments, RL spends a lot of samples with the chair quite far from the goal location. We verify this by plotting the average reward 7 for the chair tasks. The reward for this task is of the form $-x + e^{-x}$, where x is the distance of the chair to the goal position of the chair. The negative mean reward for RL without the prior implies that the distance x to the goal is quite large, meaning that the robot is often far from the goal. On the other hand, since we execute the prior and policy sequentially for the chair task, our policy always starts out reasonably close to the goal. However, the prior often fails, as can be seen from its low average success rate reported in Fig.6. In analyzing the qualitative behavior, we find that when the planner encounters a collision between the chair and table, it is unable to adapt to correct this. We depict an example of this planner failure in Fig.8. In contrast, our approach adapts the policy based on its experience.

B. Final Policy Evaluation

We evaluate the final policies obtained after training and find that our approach obtains an average success rate of 80% across tasks from Table I. For evaluation, we use the deterministic policy instead of sampling from the stochastic distribution which is used during training. Further, we set the initial state of the objects to be close to the opposite goal in the goal cycle. For instance, in the sweeping task, we initialize the paper bag roughly in the location shown in Fig.3-h. This is different from training, where the paper bag could end up in any location, and success is continually evaluated. We note that on the particularly hard task of sweeping, none of the other methods are successful, while our approach gets 80% success. The case of long-handle dustpan pickup is notable since offline RL performs on par with our method, getting about 60% success. While the numerical performance is similar, there is a considerable qualitative difference in the behavior learned. Our approach encounters reward via actions not demonstrated in the prior, and so explores those outcomes and develops strategies that are quite different from the prior. This involves raising the robot's arm and dropping the dustpan, such that it lands upright. On the other hand, offline RL sticks very close to the prior distribution and instead learns skills that involve pushing the dustpan down along with the arm. We observe that this is the only task for which offline RL is performant, getting an average success of only 13% on the remaining tasks. One explanation for why this is the case is that in the dustpan task, all the demonstrations involving the prior that succeed are similar in structure due to a lack of mobility needed for the task. Hence, it is easier for offline RL to learn successful strategies since they repeat often in the prior dataset. For chair moving and sweeping tasks, though, the successful trajectories are likely more diverse owing to the robot having to move to a target location, causing offline RL to struggle.

VI. Conclusion

We have presented an approach for continuously learning new mobile manipulation skills via autonomous real-world RL. This is enabled using the automated collection of useful data, the acquisition of diverse practices via automated resets using multi-goal or multi-robot setups, efficient real-world control using priors, and flexible reward definition. We show our approach can enable Boston Dynamics Spot robots to learn challenging mobile manipulation skills, including moving chairs, picking up long-handled dustpans, and sweeping, with an average task success rate of about 80%, showing a 3-4X improvement over previous approaches.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do As I Can and Not As I Say: Grounding Language in Robotic Affordances. In *arXiv preprint arXiv:2204.01691*, 2022. 3

[2] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving Rubik's Cube with a Robot Hand. *arXiv preprint arXiv:1910.07113*, 2019. 1

[3] Max Bajracharya, James Borders, Dan Helmick, Thomas Kollar, Michael Laskey, John Leichty, Jeremy Ma, Umashankar Nagarajan, Akiyoshi Ochiai, Josh Petersen, et al. A Mobile Manipulation System for One-Shot Teaching of Complex Tasks in Homes. In *ICRA*, pages 11039–11045. IEEE, 2020. 3

[4] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient Online Reinforcement Learning with Offline Data. In *ICML*, 2023. 5, 6

[5] Kate Baumli, Satinder Baveja, Feryal Behbahani, Harris Chan, Gheorghe Comanici, Sebastian Flennerhag, Maxime Gazeau, Kristian Holsheimer, Dan Horgan, Michael Laskin, et al. Vision-Language Models as a Source of Rewards. *arXiv preprint arXiv:2312.09187*, 2023. 3

[6] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable Transformations of Python+NumPy Programs, 2018. URL http://github.com/google/jax. 6

[7] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. RT-1: Robotics Transformer for Real-World Control at Scale. In *arXiv preprint arXiv:2212.06817*, 2022. 3

[8] Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized Ensembled Double Q-Learning: Learning Fast Without a Model. In *ICLR*, 2021. 6

[9] Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme Parkour with Legged Robots. *arXiv preprint arXiv:2309.14341*, 2023. 1

[10] Mark Cutler, Thomas J Walsh, and Jonathan P How. Reinforcement Learning with Multi-Fidelity Simulators. In *ICRA*, pages 3888–3895. IEEE, 2014. 1

[11] Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational Inverse Control with Events: A General Framework for Data-Driven Reward Definition. In *NeurIPS*, volume 31, 2018. 2

[12] Zipeng Fu, Xuxin Cheng, and Deepak Pathak. Deep Whole-Body Control: Learning a Unified Policy for Manipulation and Locomotion. In *CoRL*, 2022. 3

[13] Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile ALOHA: Learning Bimanual Mobile Manipulation with Low-Cost Whole-Body Teleoperation. In *arXiv*, 2024. 3

[14] Jiayuan Gu, Devendra Singh Chaplot, Hao Su, and Jitendra Malik. Multi-Skill Mobile Manipulation for Object Rearrangement. In *ICLR*, 2023. 3

[15] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Prakashchand Gandhi, and Lerrel Pinto. Robot Learning in Homes: Improving Generalization and Reducing Dataset Bias. In *NeurIPS*, volume 31, 2018. 3

[16] Abhishek Gupta, Justin Yu, Tony Z Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, Thomas Devlin, and Sergey Levine. Reset-Free Reinforcement Learning via Multi-Task Learning: Learning Dexterous Manipulation Behaviors without Human Intervention. In *ICRA*, pages 6664–6671. IEEE, 2021. 2, 3

[17] Abhishek Gupta, Corey Lynch, Brandon Kinman, Garrett Peake, Sergey Levine, and Karol Hausman. Demonstration-Bootstrapped Autonomous Practicing via Multi-Task Reinforcement Learning. In *ICRA*, pages 5020–5026. IEEE, 2023. 2

[18] Sehoon Ha, Peng Xu, Zhenyu Tan, Sergey Levine, and Jie Tan. Learning to Walk in the Real World with Minimal Human Effort. In *CoRL*, 2020. 3

[19] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *ICML*, pages 1861–1870, 2018. 6

[20] Tuomas Haarnoja, Ben Moran, Guy Lever, Sandy H Huang, Dhruva Tirumala, Markus Wulfmeier, Jan Humplik, Saran Tunyasuvunakool, Noah Y Siegel, Roland Hafner, et al. Learning Agile Soccer Skills for a Bipedal Robot with Deep Reinforcement Learning. *arXiv preprint arXiv:2304.13653*, 2023. 1

[21] W. Han, S. Levine, and P. Abbeel. Learning Compound Multi-Step Controllers under Unknown Dynamics. In *IROS*, 2015. 2

[22] Zheyuan Hu, Aaron Rovinsky, Jianlan Luo, Vikash Kumar, Abhishek Gupta, and Sergey Levine. REBOOT: Reuse Data for Bootstrapping Efficient Real-World Dexterous Manipulation. In *CoRL*, 2023. 5

[23] Snehal Jauhri, Jan Peters, and Georgia Chalvatzaki. Robot Learning of Mobile Manipulation With Reachability Behavior Priors. *IEEE Robotics and Automation Letters*, 7(3):8399–8406, 2022. 3

[24] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation. *arXiv preprint arXiv:1806.10293*, 2018. 1, 3

[25] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale. *arXiv preprint arXiv:2104.08212*, 2021. 1, 3

[26] 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. 3, 4, 6

[27] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline Reinforcement Learning with Implicit Q-Learning. In *ICLR*, 2022. 7

[28] Eric Krotkov, Douglas Hackett, Larry Jackel, Michael Perschbacher, James Pippine, Jesse Strauss, Gill Pratt, and Christopher Orlowski. The DARPA Robotics Challenge Finals: Results and Perspectives. *Journal of Field Robotics*, 34(2):229 – 240, 2 2017. doi: 10.1002/rob.21683. 3

[29] Steven M. LaValle and Jr. James J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001. doi: 10.1177/02783640122067453. 2

[30] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-End Training of Deep Visuomotor Policies. *JMLR*, 2016. 1, 3

[31] Hao Li, Xue Yang, Zhaokai Wang, Xizhou Zhu, Jie Zhou, Yu Qiao, Xiaogang Wang, Hongsheng Li, Lewei Lu, and Jifeng Dai. Auto MC-Reward: Automated Dense Reward Design with Large Language Models for Minecraft. *arXiv preprint arXiv:2312.09238*, 2023. 3

[32] Huihan Liu, Soroush Nasiriany, Lance Zhang, Zhiyao Bao, and Yuke Zhu. Robot Learning on the Job: Human-in-the-Loop Autonomy and Learning During Deployment. *arXiv preprint arXiv:2211.08416*, 2022. 2

[33] 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. 4

[34] Yuntao Ma, Farbod Farshidian, Takahiro Miki, Joonho Lee, and Marco Hutter. Combining Learning-Based Locomotion Policy With Model-Based Manipulation for Legged Mobile Manipulators. *IEEE Robotics and Automation Letters*, 7(2):2377–2384, 2022. 3

[35] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. ALAN: Autonomously Exploring Robotic Agents in the Real World. In *ICRA*, 2023. 2

[36] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep Dynamics Models for Learning Dexterous Manipulation. In *CoRL*, pages 1101–1112. PMLR, 2020. 3

[37] Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-Fit: State-Covering Self-Supervised Reinforcement Learning. In *ICML*, 2020. 2

[38] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *RSS*, 2018. 5

[39] C. Schenck and D. Fox. Visual Closed-Loop Control for Pouring Liquids. In *ICRA*, 2017. 2

[40] Nur Muhammad Mahi Shafiullah, Anant Rai, Haritheja Etukuru, Yiqian Liu, Ishan Misra, Soumith Chintala, and Lerrel Pinto. On Bringing Robots Home. *arXiv preprint arXiv:2311.16098*, 2023. 3

[41] Archit Sharma, Ahmed M Ahmed, Rehaan Ahmad, and Chelsea Finn. Self-Improving Robots: End-to-End Autonomous Visuomotor Reinforcement Learning. *arXiv preprint arXiv:2303.01488*, 2023. 3

[42] Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-End Robotic Reinforcement Learning without Reward Engineering. In *RSS*, 2019. 2

[43] Laura Smith, J Chase Kew, Xue Bin Peng, Sehoon Ha, Jie Tan, and Sergey Levine. Legged Robots that Keep on Learning: Fine-Tuning Locomotion Policies in the Real World. In *ICRA*, pages 1593–1599. IEEE, 2022. 3

[44] Laura Smith, Ilya Kostrikov, and Sergey Levine. Demonstrating a Walk in the Park: Learning to Walk in 20 Minutes With Model-Free Reinforcement Learning. In *RSS*, 2022. 3

[45] Laura Smith, J Chase Kew, Tianyu Li, Linda Luu, Xue Bin Peng, Sehoon Ha, Jie Tan, and Sergey Levine. Learning and Adapting Agile Locomotion Skills by Transferring Experience. *arXiv preprint arXiv:2304.09834*, 2023. 5

[46] 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. 5

[47] Charles Sun, Jedrzej Orbik, Coline Manon Devin, Brian H Yang, Abhishek Gupta, Glen Berseth, and Sergey Levine. Fully Autonomous Real-World Reinforcement Learning

with Applications to Mobile Manipulation. In *CoRL*, pages 308–319. PMLR, 2022. 3

[48] Dhruva Tirumala, Thomas Lampe, Jose Enrique Chen, Tuomas Haarnoja, Sandy Huang, Guy Lever, Ben Moran, Tim Hertweck, Leonard Hasenclever, Martin Riedmiller, Nicolas Heess, and Markus Wulfmeier. Replay across Experiments: A Natural Extension of Off-Policy RL. *arXiv preprint arXiv:2311.15951*, 2023. 5

[49] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In *IROS*, pages 23–30. IEEE, 2017. 1

[50] Bohan Wu, Roberto Martin-Martin, and Li Fei-Fei. M-EMBER: Tackling Long-Horizon Mobile Manipulation via Factorized Domain Transfer. In *ICRA*, 2023. 3

[51] Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, and Thomas Funkhouser. TidyBot: Personalized Robot Assistance with Large Language Models. *Autonomous Robots*, 2023. 3

[52] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and Pieter Abbeel. DayDreamer: World Models for Physical Robot Learning. In *CoRL*, 2023. 3

[53] Fei Xia, Chengshu Li, Roberto Martín-Martín, Or Litany, Alexander Toshev, and Silvio Savarese. ReLMoGen: Integrating Motion Generation in Reinforcement Learning for Mobile Manipulation. In *ICRA*, pages 4583–4590. IEEE, 2021. 3

[54] Haoyu Xiong, Russell Mendonca, Kenneth Shaw, and Deepak Pathak. Adaptive Mobile Manipulation for Articulated Objects In the Open World. *arXiv preprint arXiv:2401.14403*, 2024. 3

[55] Kelvin Xu, Zheyuan Hu, Ria Doshi, Aaron Rovinsky, Vikash Kumar, Abhishek Gupta, and Sergey Levine. Dexterous Manipulation from Images: Autonomous Real-World RL via Substep Guidance. In *ICRA*, pages 5938–5945. IEEE, 2023. 3

[56] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine. Collective Robot Reinforcement Learning with Distributed Asynchronous Guided Policy Search. In *IROS*, 2017. 2

[57] Ruihan Yang, Yejin Kim, Aniruddha Kembhavi, Xiaolong Wang, and Kiana Ehsani. Harmonic Mobile Manipulation. *arXiv preprint arXiv:2312.06639*, 2023. 1, 3

[58] Denis Yarats, Rob Fergus, and Ilya Kostrikov. Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels. In *ICLR*, 2021. 6

[59] Naoki Yokoyama, Alex Clegg, Joanne Truong, Eric Undersander, Tsung-Yen Yang, Sergio Arnaud, Sehoon Ha, Dhruv Batra, and Akshara Rai. ASC: Adaptive Skill Coordination for Robotic Mobile Manipulation. *IEEE Robotics and Automation Letters*, 9(1):779–786, 2024. doi: 10.1109/LRA.2023.3336109. 3

[60] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language to Rewards for Robotic Skill Synthesis. *arXiv preprint arXiv:2306.08647*, 2023. 3

[61] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Phillip Krähenbühl, and Ishan Misra. Detecting Twenty-Thousand Classes Using Image-Level Supervision. In *ECCV*, 2022. 4, 6

[62] Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine. The Ingredients of Real-World Robotic Reinforcement Learning. In *ICLR*, 2020. 3