Fully Autonomous Real-World Reinforcement Learning for Mobile Manipulation

Anonymous Author(s)
Affiliation
Address
email

Abstract: We study how robots can autonomously learn skills that require a combination of navigation and grasping. While reinforcement learning in principle provides for automated robotic skill learning, in practice reinforcement learning in the real world is challenging and often requires extensive instrumentation and supervision. Our aim is to devise a robotic reinforcement learning system for learning navigation and manipulation together, in an *autonomous* way without human intervention, enabling continual learning under realistic assumptions. Our proposed system, ReLMM, can learn continuously on a real-world platform without any environment instrumentation, without human intervention, and without access to privileged information, such as maps, objects positions, or a global view of the environment. Our method employs a modularized policy with components for manipulation and navigation, where manipulation policy uncertainty drives exploration for the navigation controller, and the manipulation module provides rewards for navigation. We evaluate our method on a room cleanup task, where the robot must navigate to and pick up items scattered on the floor. After a grasp curriculum training phase, ReLMM can learn navigation and grasping together fully automatically in around 40 hours of autonomous real-world training.

Keywords: Mobile Manipulation, Reinforcement Learning, Reset-Free

1 Introduction

Learning-based approaches have the potential to bring robots into open-world environments with end-to-end vision-based policies that can perform tasks without external instrumentation, AR-tagging, or expensive sensors. Such learning systems can be particularly beneficial for mobile manipulators, which perform tasks while navigating through open-world environments that are not practical to instrument or map out in advance. Unfortunately, in practice, most real-world reinforcement learning (RL) systems require careful environment instrumentation and human supervision or demonstrations during the training process to ensure that the robot performs effective exploration and resets between trials, making them difficult to apply to the kinds of open-world domains where we might want mobile manipulators to operate. Such systems might require specially installed infrastructure that provides explicit resets [1, 2, 3, 4], or the presence of a person providing resets and monitoring the learning process [5, 6], and cannot simply be dropped into a natural environment and continue learning.

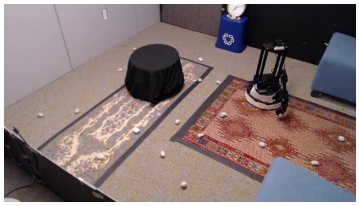

Figure 1: ReLMM enables learning mobile manipulation skills autonomously in the real world, using only on-board sensing.

To address this issue and make it possible for mobile manipulators to learn with RL directly in the real world, we propose a system for learning mobile manipulation skills without instrumentation, demonstrations, or manually-provided reset mechanisms. We aim to produce a system that enables a robot to learn autonomously in settings such as homes and offices, such that anyone could simply

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

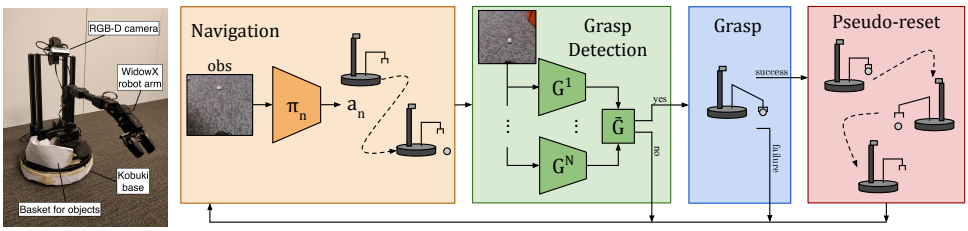

Figure 2: **Method overview.** ReLMM partitions the mobile manipulator into a navigation policy and grasping policy. Both policies are rewarded when an object is grasped. We use an ensemble of action-conditioned grasp success prediction functions estimate the success of a potential grasp better and use the uncertainty as an exploration bonus. If grasp success is likely, a grasp action is sampled from the grasp predictors and executed. If the grasp is successful during training, the robot executes a pseudo-reset by placing the collected object back down in a random location.

place the robot down, start the learning process, and return to a trained robot. This goal dictates several constraints that shape our method: (1) the robot must learn entirely from its own sensors, both to select actions and to compute rewards; (2) the entire learning process must be efficient enough for real-world training; (3) the robot must be able to continually gathering data at scale without human effort. A system that meets these requirements would not only be able to learn skills in open-world settings, but could also continue to improve throughout its lifetime: when learning can be performed practically without instrumentation, there is no reason to stop the learning process at deployment time, and the robot keep getting better and better at its given task perpetually. Our aim is not to propose the best possible system for solving any particular task. Rather, we aim to show how to create a real-world reinforcement learning system that enables learning mobile manipulation skills entirely from real-world interaction, with minimal human intervention.

Our contribution is a system for autonomously training a mobile manipulation robot that satisfies the above constraints, which we call Reinforcement Learning for Mobile Manipulation (ReLMM). We apply ReLMM to the task of collecting items scattered across a room. Our system learns directly from on-board ego-centric camera observations and uses proprioceptive grasp sensing to assign itself rewards. To ensure the learning process is sample efficient and to facilitate exploration, we split the robot's controller into separate navigation and grasping neural network policy modules that choose when to act based on their predicted value and are continuously trained together for the same objective: successfully grasping objects in the environment. Separating the policies enables the use of uncertainty-based exploration for the grasping module, which uses an ensemble of Q-functions to explore grasp actions efficiently. Lastly, to reduce the need for humans to provide interventions in the form of resets, we develop an autonomous resetting behavior where the robot re-arranges the environment as it learns, so as to continually create new arrangements of objects for the agent continually "practice." Together, these components plus a brief grasping curriculum enable a system that can operate in real-world environments, learning how to navigate and grasp from its own collected experience, and mastering room cleanup tasks with about 40-60 hours of autonomous interaction.

2 Related Work

Robotic mobile manipulation tasks pose a number of unique challenges [7]. Many prior methods have addressed these challenges by requiring human effort for instrumentation and state estimation [8, 9, 10, 11], hand-coded controllers [12], or demonstrations [10, 13, 14]. Several methods also require external instrumentation such as top-down camera views, oracle knowledge of object pose, or precomputed navigation maps [15, 16, 17]. textcolorMobile manipulation has also seen benefits from combining learning and planning for more efficient exploration in simulation [18, 19, 20], which allow for safe and effective behavior in the real world if object or goal locations are provided. While these choices are pragmatic, they do not address the problem of learning continually in uninstrumented real-world settings and require a simulated world to train in. In contrast, our proposed system is aimed at enabling reinforcement learning directly on the real hardware that is maximally autonomous, and does not require external instrumentation.

Hierarchical reinforcement learning has been shown to learn interactive navigation and object rearrangement tasks in simulation operating from first-person view [21, 22], but require millions of

timesteps to train and have not been demonstrated in the real world. Our system uses a similar hierarchical structure to these, but targets accessible real world learning by leveraging policy uncertainty estimates and curriculum learning for realistic sample efficiency.

While RL-based methods allow agents to improve via interaction, enabling robots that can learn outside of instrumented laboratory settings is difficult. Such settings lack episodic structure and well-shaped reward functions [23, 24, 25, 26, 27] that are important for success [28, 29]. Large-scale uninterrupted deployment and training in the real world has been studied in several prior works [30, 31, 32, 33, 34, 35, 36]. While many of these works train robots in the real world without human interventions, they focus on navigation without manipulation. Large-scale real-world learning has also been successful for robotic grasping in laboratory settings. By letting robots learn from their own experience, prior approaches have shown that robots can learn to grasp from images [37, 38, 39, 40] or point clouds [41, 42]. These methods show real world learning can lead to robust robotic behavior, but they do not tackle challenging mobile manipulation tasks.

Gupta et al. show an approach to training a grasping policy on a mobile manipulator, but unlike our work, do not attempt to learn to navigate [43]. Wang et al. train a mobile manipulator with RL in simulation, but require known object pose, dense rewards, and episodic resets to a single initial state [16]. In contrast, our approach learns entirely from a sparse reward and onbaord sensors. Past approaches to learning without episodic resets have focused on stationary manipulation or simulation [44, 45, 2]. In this work we explicitly focus on how *mobile manipulation* robots can learn without external sources of resets or state estimation in the real world and lay out a set of design decisions that makes this process a practical one for acquiring robot skills.

3 Preliminaries

In this work, we use reinforcement learning (RL) as a general purpose algorithm for learning robotic behaviors. Reinforcement learning has the advantage of being able to operate on autonomously collected data, and enables the robot to improve through trial and error. To this end, the mobile manipulation task is formulated as a partially observed Markov decision processes (POMDP) with an observation space $\mathcal{O}$ of first person RGB images, state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $r(s_t, a_t)$, transition dynamics $\mathcal{P}(s_{t+1}|s_t, a_t)$, observation probability $\mathcal{P}(o_t|s_t)$, a discount factor γ, and an initial state distribution $\rho(s_0)$. The goal of reinforcement learning is to learn a control policy $\pi(a_t|o_t)$ that can determine which actions to take in each observation such that the expected sum of rewards is maximized. This objective can be written as $J(\pi) = \mathbb{E}_{a_t \sim \pi(o_t), s_t, o_t \sim \mathcal{P}} \left[\sum_t \gamma^t r(s_t, a_t)\right]$, as for a standard RL problem.

4 ReLMM: RL For Mobile Manipulation

We develop the ReLMM system to enable training mobile manipulation robots with RL. While we specifically apply it to a room cleaning task, in principle ReLMM could be used to learn other mobile manipulation tasks as well. Each component of ReLMM is chosen in order to maximise the autonomy of learning while retaining the sample efficiency needed to train in the real world. The specific task that we study in our experiments involves training a robot to quickly navigate around in a room with obstacles, physically pick up many objects, and place them in a basket mounted on the robot, as shown in Figs 1 and 3.

Algorithm 1 TrainGrasp($G^1, ..., G^M$, $\mathcal{D}_g$, N, st)

for t = 0, . . . , N steps **do**
  Get grasp observation $\tilde{o}$.
  Sample $a_g \sim \pi_g(\cdot|\tilde{o})$. // see Equation 2
  Perform grasp a_g, receiving $r_g = 0$ or 1.
  Store $(\tilde{o}, a_g, r_g)$ in $\mathcal{D}_g$.
  Update $G^1, ..., G^M$ on $\mathcal{D}_g$
  if st =True, Drop object if held.
  elseif $r_g = 1$ and st =False, **return** r_g
return 0

Our system provides for efficient autonomous learning by decomposing the policy into grasping and navigation policies, using an ensemble of grasping models to explore based on uncertainty, automatically rearranging the environment after successful grasps, and using a curriculum to bootstrap and stabilize the concurrent training of both policies. Our final system can learn room cleaning skills in a number of different room configurations in ~ 40 hours directly in the real world.

4.1 Grasping Policy Training

As noted previously, we decomposed the control problem into grasping and navigation. This gives the manipulation policy two objectives: given an image observation, accurately model the robots chance of success should it attempt a grasp and, if so, select an appropriate action to maximize success. We obtain the former by training an ensemble of grasping policies and using their uncertainty to efficiently explore grasping. For choosing *how* to grasp, the policy must learn with sufficiently low sample complexity so as to make real-world training feasible. To reduce the complexity of the exploration problem we formulate grasping as a discrete single-step top-down action. The grasp policy $\pi_g(a_g|o_g)$ is parameterized with the action a_g discretized in the x-y plane, and the observation o_g corresponding to a RGB image from the robot's camera. Such single-step action selection formulations are amenable to more efficient training than more complex multi-step tasks [46, 47].

Framed in this way, the grasping task corresponds to a contextual multi-armed bandit problem. Specifically, we train grasping policies that, given an image, predict the likelihood of grasp success for each action.We use a soft-max over the action values to sample actions in proportion to their exponentiated probability of success. To create the ensemble, we train $M = 6$ independent grasp policies, $G^1 \dots G^M$ that are each by minimizing the cross-entropy loss on the same dataset:

$$\mathcal{L}_g^i = \mathbb{E}_{(o_g, a_g, r_g) \sim \mathcal{D}_g}[-r_g \log G^i(o_g, a_g) - (1 - r_g) \log(1 - G^i(o_g, a_g))]. \tag{1}$$

Here, r_g is 1 when the robot successfully grasps an object, which is determined by presenting the gripper to the onboard camera. The grasping exploration policy is formed by constructing a Boltzmann distribution from optimistic estimates of grasp success, where the mean estimate from the ensemble, $\mathbb{E}[G^i(o_g, a_g)]$, is modified by adding a multiple of the ensemble variance $\sigma(G^i(o_g, a_g))$, which we expect to be larger for actions where success is more uncertain:

$$\tilde{G}(o_g, a_g) = \alpha \mathbb{E}[G^i(o_g, a_g)] + \beta \sigma(G^i(o_g, a_g)), \tag{2}$$

with $\pi_g(a_g|o_g) \propto \exp(\tilde{G}(o_g, a_g))$. The expectation and standard deviation are taken over the ensemble, and $\alpha, \beta \geq 0$ are hyperparameters. Other equivalent multi-step off-policy or contextual bandit algorithms can be used to train the grasping policy. However, we show in Section 6.3 that they are not as sample efficient for the mobile manipulation task in this work. Algorithm 1 describes the grasp training process in further detail.

4.2 Navigation Policy Training

For navigation, the policy must be able to control the mobile base to approach objects in a way that the current grasping policy can succeed. The navigation policy $\pi_n(a_n|o)$ outputs the action a_n that controls the forward and turn velocities of the mobile robot base.

At every time step, the agent has to decide whether to perform a grasp or not by balancing the opportunity of receiving reward, the chance to collect novel data for the grasping policies, and the cost of wasting a timestep if there is no object within reach. This balancing act is done by reusing the same uncertainty measure described in Equation 2 when choosing whether to attempt a grasp. The probability of whether to attempt a grasp is

$$\mathbb{P}[\text{grasp}|o] = \max_{a_g} \tilde{G}(o, a_g). \tag{3}$$

Under this design, the navigation policy $\pi_n(a_n|o)$ continues to experience observations and output navigation actions, and at every step the choice of whether to perform a grasp or not is made by sampling from grasp success probability of the current grasp model $\mathbb{P}[\text{grasp}|o]$ defined in Eqn 3. When the model decides attempting a grasp is worth the risk, the robot executes a grasp and evaluates the outcome to provide the navigation agent a reward. From the perspective of the navigation policy, the choice of whether to grasp or not is a part of the inherent dynamics of the environment. We compare two possible rewards r_n for the the navigation policy. The first option is directly optimizing for the task by rewarding the navigation when a grasp is successful (i.e. where the current grasp ensemble has high performance): $r_n(o) = r_g(o) - 1$. The second option is to reward the navigation for reaching states that the current grasp ensemble will choose to grasp at, which is a function of its mean and uncertainty: $r_n(o) = (\mathbb{P}[\text{grasp}|o] - 1)$ that we use to *relabel* states without successful grasps during SAC policy update. As this reward function only depends on the grasp ensemble and not on actual grasp success, this reward is computed at every policy update step and

in Figure 4 we show how it can improve sample efficiency. The RL objective for navigation is $\max_{\pi_n} \mathbb{E}_{\pi_n(a_n|o_t)} \left[\sum_{k=0}^{\infty} \gamma^k r_n(o_{t+k})\right]$ We train the navigation policy for this objective using soft actor critic (SAC) [48, 49].

4.3 Training with Autonomous Pseudo-Resets

While the training schemes described above allow ReLMM to learn efficiently, both the contextual bandit grasping formulation and the navigation training setup requires an episodic training setup, where the environment is reset between trials, for example by replacing the objects into the world at random positions. To enable the robot to learn mobile manipulation skills without manually provided resets, we construct an automated pseudo-reset system that allows our method to learn autonomously without human intervention. After a successful grasp, the environment would ideally be reset by relocating the object to a new, randomly selected location. In stationary bin grasping setups, this can be automated simply by dropping the object back into the bin. However, for a mobile manipulator, dropping the object back to where it was grasped would promote policies that fail to search for new objects, and simply remain in the same location. To force the policies to learn to grasp in varied situations, we perform an automated random pseudo-reset, by commanding random navigation actions while the robot is holding the object, placing down the object in this new location, and then navigating randomly away. This ensures the robot will not always be near an object during training and must instead learn to seek objects out. We outline the overall algorithm used for training in Algorithm 2.

Algorithm 2 ReLMM (with Stationary Curriculum)

Hyperparameters: M, N_{st}, N_{grasp}, *relabel*
Init: function estimators π_n, $G^1, \ldots, G^M$.
Replay buffers $\mathcal{D}_n = \{\}$, $\mathcal{D}_g = \{\}$
TrainGrasp($G^1, .., G^M$, D_g, N_{st}, True)
for $t = 0, \ldots, T$ steps **do**
  Get navigation observation o_t
  Sample $a_n \sim \pi_n(\cdot|o_t)$ and perform a_n
  if *uniform*() $\leq \mathbb{P}[\text{grasp}|o_t]$ **then**
    r_g =TrainGrasp(G^1,.., G^M,D_g,N_{grasp},False)
  else $r_g = 0$
  if *relabel* **then**
    Navigation reward $r_n = \mathbb{P}[\text{grasp}|o_t] - 1$
  else $r_n = r_g - 1$
  Get next navigation observation o_{t+1}
  Store (o_t, a_n, r_n, o_{t+1}) in $\mathcal{D}_n$.
  Update π_n with $\mathcal{D}_n$ using SAC.
  Pseudo-reset
end for

4.4 Training Curricula

The separate grasping and navigation policies lend themselves naturally to curricular training, where the grasping policy, which needs to have some successes to provide rewards to the navigation policy, can be prioritized at the beginning of the learning process. We propose two types of curricula, which both break the potential "chicken-and-egg" training problem of providing poor reward signals for navigation from an untrained grasping model.

Stationary curriculum. The simplest curriculum is to place a single object in front of the robot and run Algorithm 1 with $st =$ True for $N_{\text{st}} = 2000$ steps. After each successful grasp, the robot places the object down randomly. If the object is knocked into an area the robot can not reach (because the base is stationary), which happens about 5% of the time, a human observer must push the object back into the graspable area. This curriculum is very time efficient, but does require occasional human intervention.

Autonomous curriculum. For fully autonomous learning, we develop a training curriculum that favors collecting grasping data early on by performing a high number of grasps at the beginning of training, according to hyperparameters N_{start}, N_{stop}, and N_{max}. The robot attempts $N_{grasp} = N_{start}$ grasps after every navigation step. If the robot succeeds at grasping an object it will practice with this object until $N_{grasp} = N_{stop}$ unsuccessful grasps. This initial automatic grasping curricula ends when a total of N_{max} grasps are complete. More details on the grasping curriculum algorithm are in Appendix B.

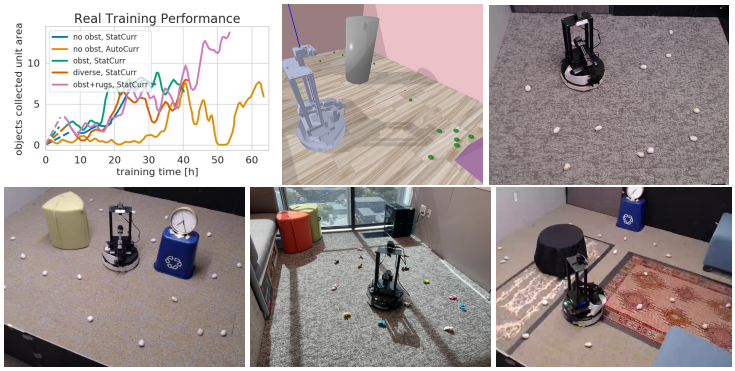

Figure 3: **Top Left**: Learning curves for training the real robot showing the number of objects grasping in the last 15 min of operation, which is ~ 250 actions. This metric are divided by the rooms' surface areas to make them comparable. **Top Center**: The simulated environment, which we run with and without obstacles. **Right & Bottom**: Snapshots from the evaluation tasks without obstacles in a $4m^2$ room (top-right, *no obst*, bottom-center, *diverse*), with obstacles *obst* in a $9m^2$ room (bottom-left), and with obstacles and rugs *obst+rugs* in a $10m^2$ room (bottom-right).

5 Robotic Mobile Manipulator: System Overview

Our choice of robotic system reflects the need for a robust robotic platform that can operate autonomously for long periods of time, and is unlikely to cause damage to itself and its surroundings. Ensuring safety during autonomous operation is itself a significant research challenge, which is outside of the scope of this work. Therefore, we utilize a small-scale low-cost mobile manipulation platform based on the LocoBot design [43, 50], shown in Figure 2 (left), which consists of an iClebo Kobuki mobile base and a WidowX200 5-DoF arm. The robot sensors include an Intel RealSense D435 camera at the top of the robot, as well as bump sensors on the base. We use an onboard Intel NUC to command the robot, and connect wirelessly to a server for data processing and training. Random exploration is generally safe with the LoCoBot as it is small, light, and will automatically stop the arm's motors if they encounter resistance. We also use the depth camera output to automatically avoid collisions, as described in subsection A.3. The robot learns in a real-world office space, with varied lighting conditions, distractors, and surface textures. Our experiments utilize small objects that the robot can feasibly pick up, which consist of socks and toys – a sampling of objects one may want a robot to pick up off the floor.

To control the robot, we separately command the iClebo mobile base and the WidowX200 robotic arm with the corresponding navigation and a grasping policies. The navigation policy stops the robot during a grasp. For grasping, we use a directed end-effector control space. Assuming the floor is flat, the vertical position for grasping is always chosen to be just above the floor, and the learning algorithm chooses a point in $X - Y$ space in front of the robot to perform a grasp at. This chosen position then dictates where to move the gripper using inverse kinematics. Details on the learned network architectures are in Appendix A.2.

6 Experimental Results

Our experiments aim to evaluate our autonomous reinforcement learning system in a number of real-world environments, as well as to provide ablation experiments and analysis in simulation. In particular, we aim to study the following questions: (1) Can ReLMM learn autonomously in the real world? (2) How does the control hierarchy affect learning performance? (3) How does ReLMM compare to other policy designs and prior methods? Our goal is to study real-world reinforcement learning, rather than necessarily provide the best possible solution to the room clearing task, and hence we design our system to be general, with only the reward determining the task.

6.1 Experiment details

For all real-world experiments, the entire training procedure is performed in the real world on the LoCoBot platform, with about 25 to 50 hours of training depending on the environment. The training

is autonomous, with the exception of the stationary curriculum (if used), and battery changes every ~ 5 hours, during which time we may replace objects that become stuck near corners, or walls.

As our task is non-episodic, the policies are evaluated at the end of training. This is done by scattering the objects, executing the policy (without pseudo-resets) for 15 minutes, and counting the number of objects collected in that time. We use four real environments in our evaluation (no obstacles, with obstacles, with diverse objects, and with obstacles and rugs and a simulated environments, as shown in Figure 3. Each room has a different size and number of objects, so we report the percent of objects collected in each 15 minute block of training, as shown in Figure 3. In simulation, we plot the evaluation performance using 250 timesteps instead of 15 minutes.

We compare our approach to several prior methods and baselines. The baselines include: **Rand all**, a policy that selects navigation and manipulation actions randomly, as a lower baseline; **Rand nav**, a method similar to Gupta et al. [43] that collects grasping data in the same way as our method, but has a random navigation controller. These baselines disentangle the benefits we get from learning both the grasping and the navigation. We also include a hand-coded controller (**Scripted**) specifically engineered for this task that locates objects by thresholding the image pixels and grasps at their centroids, which provides a strong hand-designed baseline. For more detail on the implementation of these baselines, see Appendix A.4.

6.2 Real Robot Evaluation

Our real-world experiments evaluate how well ReLMM can learn a mobile grasping policy autonomously in a variety of rooms. We conducted separate experiments for each of the rooms shown in Figure 3, which differ in terms of size, furniture, layout, and objects. ReLMM can train using both the *stationary* and *autonomous* grasping curricula, with the *autonomous curriculum* requiring less human effort, at the cost of increased training time. The result are summarized in Table 1 with the training time given in Table 2. Examples of the learned behaviors are also shown in Figure 6, and are illustrated in more detail in the supplementary video.

Env	no obst	obst	diverse	obst+rugs
ReLMM-StatCurr	$\mathbf{88 \pm 2}$	$\mathbf{93 \pm 2}$	$\mathbf{63 \pm 8}$	$\mathbf{78 \pm 6}$
ReLMM-AutoCurr	77 ± 8	–	–	–
Scripted	75 ± 4	88 ± 6	56 ± 3	65 ± 9
Rand nav [43]	52 ± 12	38 ± 10	22 ± 9	20 ± 7
Rand all	12 ± 6	2 ± 2	2 ± 3	5 ± 4

Table 1: Percentage of objects that the robot collects during eval in each environment (shown in Figure 6) (higher is better). Each method is trained once per env, and evaluated 3 times. The numbers are mean and stddev of the 3 evaluations. ReLMM outperforms the baselines by learning both grasping and navigation jointly, without requiring environment instrumentation. Due to ReLMM-AutoCurr's slower learning, we only evaluate it in *no obst* with *relabel*. In Table 2 we provide the training time in each env.

To provide context for these results, we compare ReLMM to the previously described baselines. The *Rand all* and *Rand nav* [43] baselines perform poorly, since neither method learns a directed navigation behavior, making it difficult to effectively navigate the room to collect objects, though *Rand nav* [43] is effective at picking up the objects if it stumbles upon them. This illustrates the importance of an intelligent navigational strategy in these settings. The *Scripted* baseline, which was specifically engineered for this task, performs reasonably in most settings, but still falls short of the policy learned by ReLMM in all of the environments, and crucially cannot improve from these results autonomously. This is particularly important in the *diverse* and *obst+rugs* environments where it is difficult to find a pixel threshold that works for all objects and all backgrounds, ReLMM can automatically learn how to identify objects to grasp via interaction. In Figure 5, we plot the evaluation performance of our ReLMM-StatCurr agent learning in *obst+rugs* for a few checkpoints, showing that our method is improving from its experience and is still improving after 50 hours of training. This is important in open-world settings, where our approach would enable mobile manipulators to improve perpetually over the lifetime of their deployment.

6.3 Simulation Analysis

To study questions (2) and (3), we perform a detailed ablation analysis in simulation. As discussed in Appendix D, we find that a discretized grasping policy learns with $\frac{1}{3}$ the samples compared to a continuous one. In Figure 4, we show the performance of various ablations of our system in the

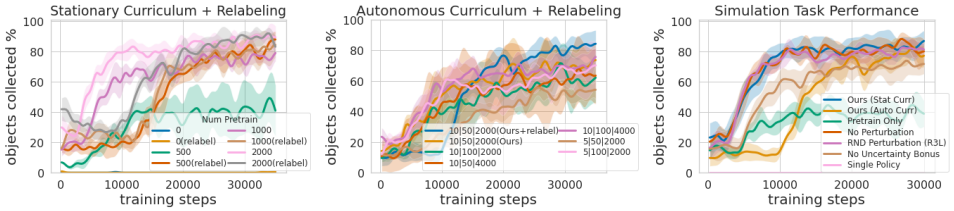

Figure 4: **Performance in the simulated environment without obstacles.** Each plot is the mean and stddev over 3 random seeds. **Left**: Ablation of different number of grasp "pretraining" samples for the stationary curriculum (N_{st}) and navigation reward relabeling (described in subsection 4.2) in simulation. Increased pretraining of the grasping policy improves overall mobile manipulation performance. **Center**: Analysis to find the best parameters for the *autonomous* curriculum compared to the *stationary* curriculum. With the proper settings ($N_{start}|N_{stop}|N_{max}$) the *autonomous* curriculum can be almost as efficient as the *stationary*, without requiring as much manual effort. **Right**: Ablation of ReLMM, all use the stationary curriculum and no relabeling except for *Ours (AutoCurr)*. We find that the uncertainty bonus and joint training are critical components of our system.

simulated environment with the *stationary curriculum*. First, we ablate the policy decomposition by training a *single policy* with reinforcement learning using grasp success as the reward. The joint action space poses a much larger exploration problem, and the policy is unable to make headway on the task in a reasonable number of samples. This indicates that the hierarchical decomposition used by ReLMM significantly improves training performance. Next we see that freezing the grasp policy after the stationary phase (*Pretrain Online*) and only training the navigation is much worse than training both together. This illustrates the interplay between the two policies, as the navigation is dependent on the grasping for obtaining rewards. As shown in the plot, training the grasp policy without the uncertainty bonus (i.e. $\beta = 0$) leads to significantly poorer performance for ReLMM, as it is less incentivized to explore. Finally, we see that the autonomous curriculum with reward relabeling can get similar final performance as the stationary curriculum with less human intervention at the cost of longer training time.

Lastly, we examine the choice of curriculum. While the stationary curriculum performs well with 2000 stationary grasps (Figure 4 left), the stationary period can require human supervision to move the objects back to the robot. The autonomous curriculum instead bootstraps the grasp learning by attempting many more grasps at the beginning of the navigation training whenever an object has been picked up. This requires no human involvement, but trains more slowly. After tuning the *autonomous* curriculum with reward relabeling in Figure 4 (center) it learn almost as fast as *stationary* in the simulated environment without obstacles. Further comparisons in other simulated environments can be found in Fig 9. Using either the stationary or autonomous with relabeling curriculum, ReLMM is able to train its policies concurrently to continue improving in realistic learning scenarios.

7 Discussion and Future Work

We presented ReLMM, a system for autonomously learning mobile manipulation skills in the real world, without instrumentation, and with minimal human intervention. Our real-world experiments, conducted in four separate rooms, show that ReLMM can train continuously for several days (40-60 hours) with only occasional interventions, and that the resulting policies are effective at cleaning up the room. Furthermore, our experiments show that ReLMM continues to improve with more training, suggesting that it provides an effective approach for lifelong learning for robotic systems deployed in open-world settings. Our simulated ablation studies further provide support for the design decisions in ReLMM, indicating that the hierarchical decomposition of navigation and grasping greatly improves learning performance, autonomous practicing with a suitable exploration bonus enhances learning speed, and our automated curriculum can provide effective performance when learning from scratch. Even when using the manually-provided curriculum, the grasp pretraining phase does not need to be long: only 1000 attempted grasps were needed to get reasonable room-cleaning performance, although more pretraining can improve performance further. Extending this system to more complex manipulation tasks would be an exciting direction for future work. We

hope that this system for practical RL on mobile manipulators will make mobile manipulation more accessible to outside lab spaces and to users who need maximally autonomous learning algorithms.

References

[1] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine. Path integral guided policy search. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3381–3388. IEEE, 2017.

[2] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine. The ingredients of real world robotic reinforcement learning. In *International Conference on Learning Representations*, 2019.

[3] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112, 2019.

[4] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in neural information processing systems*, pages 5074–5082, 2016.

[5] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine. Combining model-based and model-free updates for trajectory-centric reinforcement learning. pages 703–711, 2017.

[6] A. Ghadirzadeh, A. Maki, D. Kragic, and M. Björkman. Deep predictive policy training using reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2351–2358. IEEE, 2017.

[7] T. Nishida, Y. Takemura, Y. Fuchikawa, S. Kurogi, S. Ito, M. Obata, N. Hiratsuka, H. Miyagawa, Y. Watanabe, F. Koga, et al. Development of outdoor service robots. In *2006 SICE-ICASE International Joint Conference*, pages 2052–2057. IEEE, 2006.

[8] M. Stilman and J. J. Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. *International Journal of Humanoid Robotics*, 2(04):479–503, 2005.

[9] F. Stulp, A. Fedrizzi, and M. Beetz. Learning and performing place-based mobile manipulation. In *2009 IEEE 8th International Conference on Development and Learning*, pages 1–7, June 2009.

[10] F. Stulp, A. Fedrizzi, F. Zacharias, M. Tenorth, J. Bandouch, and M. Beetz. Combining analysis, imitation, and experience-based learning to acquire a concept of reachability in robot mobile manipulation. In *2009 9th IEEE-RAS International Conference on Humanoid Robots*, pages 161–167, Dec. 2009.

[11] L. P. Kaelbling and T. Lozano-Pérez. Unifying perception, estimation and action for mobile manipulation via belief space planning. In *2012 IEEE International Conference on Robotics and Automation*, pages 2952–2959. IEEE, 2012.

[12] P. Lehner, S. Brunner, A. Domel, H. Gmeiner, S. Riedel, B. Vodermayer, and A. Wedler. Mobile manipulation for planetary exploration. *2018 IEEE Aerospace Conference*, pages 1–11, 2018. doi:10.1109/AERO.2018.8396726.

[13] M. R. Walter, Y. Friedman, M. Antone, and S. Teller. One-shot visual appearance learning for mobile manipulation. *Int. J. Rob. Res.*, 31(4):554–567, Apr. 2012.

[14] T. Welschehold, C. Dornhege, and W. Burgard. Learning mobile manipulation actions from human demonstrations. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3196–3201, Sept. 2017.

[15] B. S. Zapata-Impata, V. Shah, H. Singh, and R. W. Platt. Autotrans: an autonomous open world transportation system. *CoRR*, abs/1810.03400, 2018. URL http://arxiv.org/abs/1810.03400.

[16] C. Wang, Q. Zhang, Q. Tian, S. Li, X. Wang, D. Lane, Y. R. Petillot, and S. Wang. Learning mobile manipulation through deep reinforcement learning. *Sensors*, 20(3):939, 2020. doi:10.3390/s20030939. URL https://doi.org/10.3390/s20030939.

[17] J. Wu, X. Sun, A. Zeng, S. Song, J. Lee, S. Rusinkiewicz, and T. Funkhouser. Spatial Action Maps for Mobile Manipulation. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi:10.15607/RSS.2020.XVI.035.

[18] J. Kindle, F. Furrer, T. Novkovic, J. J. Chung, R. Siegwart, and J. Nieto. Whole-body control of a mobile manipulator using end-to-end reinforcement learning. *arXiv preprint arXiv:2003.02637*, 2020.

[19] M. Mittal, D. Hoeller, F. Farshidian, M. Hutter, and A. Garg. Articulated object interaction in unknown scenes with whole-body mobile manipulation. *arXiv preprint arXiv:2103.10534*, 2021.

[20] D. Honerkamp, T. Welschehold, and A. Valada. Learning kinematic feasibility for mobile manipulation through deep reinforcement learning. *IEEE Robotics and Automation Letters*, 6: 6289–6296, 2021.

[21] C. Li, F. Xia, R. Martín-Martín, and S. Savarese. Hrl4in: Hierarchical reinforcement learning for interactive navigation with mobile manipulators. In *Conference on Robot Learning*, pages 603–616. PMLR, 2020.

[22] F. Xia, C. Li, R. Martín-Martín, O. Litany, A. Toshev, and S. Savarese. ReLMoGen: Leveraging motion generation in reinforcement learning for mobile manipulation. Aug. 2020.

[23] M. Sridharan and P. Stone. Structure-based color learning on a mobile robot under changing illumination. *Autonomous Robots*, 23(3):161–182, 2007.

[24] T. Krajník, J. P. Fentanes, M. Hanheide, and T. Duckett. Persistent localization and life-long mapping in changing environments using the frequency map enhancement. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4558–4563. IEEE, 2016.

[25] J. Biswas and M. M. Veloso. Localization and navigation of the cobots over long-term deployments. *I. J. Robotics Res.*, 32(14):1679–1694, 2013. doi:10.1177/0278364913503892. URL https://doi.org/10.1177/0278364913503892.

[26] S. Rosenthal, M. Veloso, and A. K. Dey. Learning accuracy and availability of humans who help mobile robots. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[27] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun. Towards robotic assistants in nursing homes: Challenges and results. *Robotics and autonomous systems*, 42(3-4):271–281, 2003.

[28] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 58:52–68, 2020.

[29] J. D. Co-Reyes, S. Sanjeev, G. Berseth, A. Gupta, and S. Levine. Ecological reinforcement learning. *CoRR*, abs/2006.12478, 2020. URL https://arxiv.org/abs/2006.12478.

[30] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial intelligence*, 128(1-2):99–141, 2001.

[31] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In J. Hendler and D. Subramanian, editors, *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence, July 18-22, 1999, Orlando, Florida, USA*, pages 343–349. AAAI Press / The MIT Press, 1999. URL http://www.aaai.org/Library/AAAI/1999/aaai99-050.php.

[32] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5129–5136. IEEE, 2018.

[33] S. A. Scherer, S. Singh, L. Chamberlain, and M. Elgersma. Flying fast and low among obstacles: Methodology and experiments. *I. J. Robotics Res.*, 27(5):549–574, 2008. doi:10.1177/0278364908090949. URL https://doi.org/10.1177/0278364908090949.

[34] J. Bruce, N. Sünderhauf, P. Mirowski, R. Hadsell, and M. Milford. Learning deployable navigation policies at kilometer scale from a single traversal. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research*, pages 346–361. PMLR, 2018. URL http://proceedings.mlr.press/v87/bruce18a.html.

[35] A. Loquercio, A. I. Maqueda, C. R. del-Blanco, and D. Scaramuzza. Dronet: Learning to fly by driving. *IEEE Robotics Autom. Lett.*, 3(2):1088–1095, 2018. doi:10.1109/LRA.2018.2795643. URL https://doi.org/10.1109/LRA.2018.2795643.

[36] G. Kahn, P. Abbeel, and S. Levine. Badgr: An autonomous self-supervised learning-based navigation system. *IEEE Robotics and Automation Letters*, 6(2):1312–1319, 2021.

[37] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[38] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016.

[39] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

[40] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In A. Billard, A. Dragan, J. Peters, and J. Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 651–673. PMLR, 29–31 Oct 2018. URL http://proceedings.mlr.press/v87/kalashnikov18a.html.

[41] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt. Grasp pose detection in point clouds. *The International Journal of Robotics Research*, 36(13-14):1455–1473, 2017.

[42] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. Aparicio, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. In *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts, July 2017. doi:10.15607/RSS.2017.XIII.058.

[43] A. Gupta, A. Murali, D. Gandhi, and L. Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 9112–9122. Curran Associates Inc., Red Hook, NY, USA, 2018.

[44] B. Eysenbach, S. Gu, J. Ibarz, and S. Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

[45] W. Han, S. Levine, and P. Abbeel. Learning compound multi-step controllers under unknown dynamics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6435–6442, 2015. doi:10.1109/IROS.2015.7354297.

[46] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan. Is q-learning provably efficient? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/d3b1fb02964aa64e257f9f26a31f72cf-Paper.pdf.

[47] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel. Learning to Manipulate Deformable Objects without Demonstrations. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi:10.15607/RSS.2020.XVI.065.

[48] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[49] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine. Learning to walk via deep reinforcement learning. In *Robotics: Science and Systems*, 2019.

[50] A. Murali, T. Chen, K. V. Alwala, D. Gandhi, L. Pinto, S. Gupta, and A. Gupta. Pyrobot: An open-source robotics framework for research and benchmarking. *CoRR*, abs/1906.08236, 2019. URL http://arxiv.org/abs/1906.08236.

A Algorithm Details

A.1 Simulated Robot

Although our main experiments are conducted entirely in the real world, without any simulation, we also constructed a simulated environment to systematically evaluate and compare various algorithmic choices and ablations of our method. We construct a simulation of our training setup using the PyBullet physics simulator. The workspace is a 3-by-3 meter room with flat walls and wooden floors. The objects consist of 20 green spheres scattered randomly across the room without obstacles, or 30 spheres in the room with obstacles. We evaluate our method both with and without obstacles (shown in Figure 3 and 8(right)). The simulated robot model is replicated to the real world and simulates how the real robot operates. During training, the environment is not reset and the same autonomous pseudo-resetting mechanism is used during training. During evaluation, the agent uses a greedy argmax policy to choose actions.

A.2 Model Architectures

For the navigation policy, the observations are 100×100 RGB images from the front facing camera, which goes into three convolutional layers of sizes 64, kernel sizes 3, and stride 1, with an average pooling layer of size 2 in between each layer. The result is passed through two dense layers of size 512. The grasping policy uses the same observations, but center cropped to 60×60 to remove unreachable regions from the image. Each of the $N = 6$ networks in the grasping ensemble is also structured as three convolutional layers of size 64, kernel size 3, and stride 2, then two dense layers of size 512. In all experiments, we discretize the grasp action space into a 15×15 grid and use $\alpha = 10$ and $\beta = 10$ for the grasping policy, $N_{\text{grasp}} = 2$, and a learning rate of 3^{-4} for training the networks.

A.3 Collision Avoidance

To ensure that we can continue training the robot for extended periods of time, the robot must be able to safely navigate in environments with obstacles without breaking. To support safe navigation, we use the depth channel of the robot's camera to detect obstacles and walls in front of the robot. When an obstacle is detected as being inside the graspable area of the robot's arm, the robot's base rotates in a random direction until the obstacle is no longer obstructing the grasp space. We also simulate this setup for more exhaustive comparisons, with details of the simulation provided in Appendix A.1.

A.4 Real Evaluation Details

Here we include additional details on the evaluations, control policies and implementation for the comparison methods.

ReLMM Due to the real world training contraints, we cannot run the evaluation protocol regularly during real-world training. We did run evaluation for several checkpoints for the ReLMM-StatCurr agent in the obst+rugs room, as shown in Figure 5. The total number of hours trained in each environment is given in Table 2. Frames from the evaluations are shown in Figure 6.

Env	no obst	obst	diverse	obst+rugs
ReLMM-StatCurr	25.1	36.2	35.7	50.9
ReLMM-AutoCurr	63.5	–	–	–

Table 2: Number of hours ReLMM was trained to achieve the performance noted in Table 1.

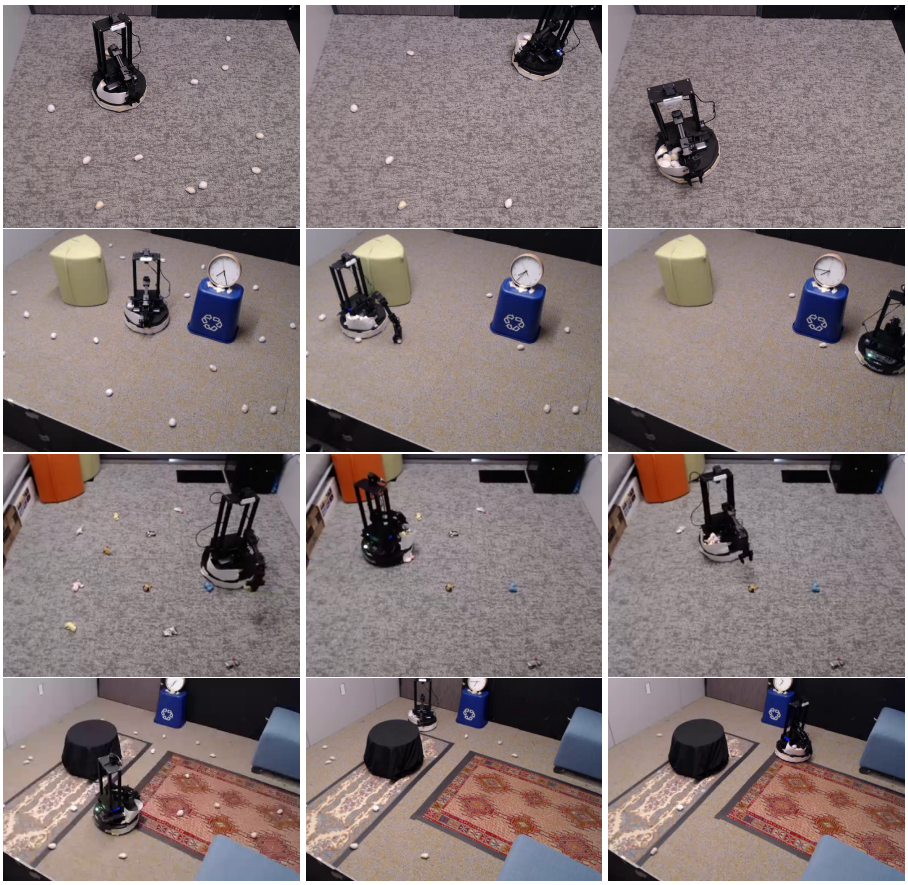

Figure 6: Frames from videos of the real robot evaluation in each room. The rooms starting from top down are *no obst*, *obst*, *diverse* and *obst+rugs*.

Rand all The navigation policy generates actions from $\pi_n \sim \mathcal{U}[-1, 1]^2$, and similarly the grasping policy chooses uniformly random from the space of discrete actions.

Rand nav [43] The navigation policy is $\pi_n \sim \mathcal{U}[-1, 1]^2$, but the grasping policy is loaded from a run of stationary curriculum right after the initial stationary phase (i.e. with stationary pretraining only).

Scripted The scripted controller takes the camera observation, convert it to grayscale, then detects contours using OpenCV and uses the contour centroids as object locations. Then it projects the pixel locations onto the flat ground plane. For grasping, it simply grasps at the closest position. For navigation, it outputs the forward and turn amount proportional to how much it needs to reach the closest object position, clipped by the action range, or random movement if there are no object in the observation.

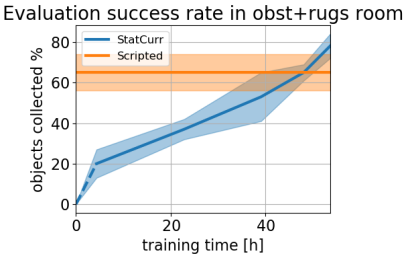

Figure 5: Different checkpoints of ReLMM in the obst+rugs room. Performance improves steadily with more training, and indeed is still increasing even at our final checkpoint, suggesting that lifelong training enables continual improvement for ReLMM.

```
def get_centroids(obs):
    gray = cv2.cvtColor(obs, cv2.COLOR_RGB2GRAY)
    _, binary = cv2.threshold(gray, 175, 255, cv2.THRESH_BINARY)
    contours, hierarchy = cv2.findContours(binary, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
    centroids = []
    for c in contours:
        M = cv2.moments(c)
        if M["m00"] == 0:
            continue
        cX = int(M["m10"] / M["m00"])
        cY = int(M["m01"] / M["m00"])
        centroids.append((cX, cY))
    return centroids

rgb_coords = np.array([
    [70., 84.],
    [45., 83.],
    [65., 83.],
    [67., 55.],
    [45., 56.],
    [56., 56.],
    [68., 70.],
    [45., 70.],
    [56., 70.],
])
robot_coords = np.array([
    [0.3,  -0.08],
    [0.3,   0.08],
    [0.3,   0.0],
    [0.47, -0.08],
    [0.47,  0.08],
    [0.47,  0.0],
    [0.38, -0.08],
    [0.38,  0.08],
    [0.38,  0.0],
])
rgb_coords = PolynomialFeatures(2).fit_transform(rgb_coords)
transmatrix = LinearRegression(fit_intercept=False).fit(rgb_coords, robot_coords).coef_.T

def get_world_from_pixel(pixel):
    rgb_coords = np.array([[pixel[0], pixel[1]]])
    rgb_coords = PolynomialFeatures(2).fit_transform(rgb_coords)
    robot_coords = rgb_coords @ transmatrix
    return robot_coords

def nav_policy(obs):
    centroids = get_centroids(obs)
    if len(centroids) == 0:
        return np.random.uniform(-1.0, 1.0, size=(2,))
    positions = [get_world_from_pixel(c) for c in centroids]
    closest = min(positions, key=lambda p: p[0] ** 2 + p[1] ** 2)
    radius = np.clip(np.sqrt(closest[0] ** 2 + closest[1] ** 2), 0.0, 0.1)
    theta = np.clip(np.arctan2(closest[1], closest[0]), -np.pi / 12, np.pi / 12)
    a0 = radius / 0.1 * 2.0 - 1.0
    a1 = theta / (np.pi / 12)
    return np.array([a0, a1])

def grasp_policy(obs):
    def filter_positions(positions):
        filtered = []
```

```
        for pos in positions:
            if 0.3 <= pos[0] <= 0.47 and -0.08 <= pos[1] <= 0.08:
                filtered.append(pos)
        return filtered
    centroids = get_centroids(obs)
    positions = [get_world_from_pixel(c) for c in centroids]
    positions = filter_positions(positions)
    if len(positions) == 0:
        return None
    closest = min(positions, key=lambda p: p[0] ** 2 + p[1] ** 2)
    return closest
```

B Grasping Curriculum

In Algorithm 3 and 4 we show the psuedocode of our Autonomous Curriculum algorithm. For our training, we also use hyperparameter values $N_{start} = 10$, $N_{stop} = 50$, and $N_{max} = 2000$. In addition to what was written in section 4.4, we include an additional hyperparameters for more stable training, $N_{bt} = 300$, which determines how many grasp there needs to be in the buffer $\mathcal{D}_g$ before grasp training begins.

C Additional Tasks

Our main experiments use the room cleanup task, but the basic design of the ReLMM system can also be extended to other tasks, as we discuss in this section. Using the simulation environment, we evaluate ReLMM on a picking and placing task, which requires picking up objects and placing them on a red rectangle (see Figure 7). The policy is trained in a similar fashion as the grasping policy, with a sparse reward, and uses the same action space. It is given reward of 1 if the object is successfully placed on one of the designated areas marked red, shown in Figure 7, and 0 otherwise. During training, we strictly follow the Algorithm 2 and sample the actions according to the distribution defined in Equation 2. In step 9 of the algorithm the decision of whether to use the grasping or placing policy depends on whether or not an object is already held by the end effector. In contrast to the grasping task, the placing task does not use the ReLMM pseudo-reset mechanism, and instead simply learns to place objects down at random positions in the environment. Analogously to the pretraining procedure we use for the grasping task, we pretrain the grasping and placing policies with $N_{\text{grasp,st}}, N_{\text{place,st}} = 2000$ samples, which matches the number used for pretraining the original grasping task. N_{obj} is the number of objects available for grasping (20 and 40), and we use $N_{\text{targets}} = 3$ random targets for placing. Since the placing portion of the task is easier compared to the more demanding grasping task, we achieve 71% success rate during placing pretraining compared to 60% for the grasping. During the combined training process, the navigation policy is given a reward -1 when not holding an object, -0.5 if the robot is holding the successfully grasped object, and a reward of 0 when the object is successfully placed on a red target, which also terminates the episode. The positions of the target areas are randomized after each successful place. This experiment illustrates that the ReLMM system is general enough to be adapted to other tasks with more complex reward structures.

D Additional Results

Here we include additional results for the paper that include images from the real robot experiments and add to the ablation analysis.

D.1 Additional Simulation Results

In simulation we study three additional conditions for ReLMM. Starting with a comparison of using a *discrete* vs *continuous* action representation. In this experiment the action space uses the same underlying control structure of finding the best X - Y position to grasp on the ground, only the output policy distribution changes. The results in Figure 8 show that training with a discrete policy trains more than twice as fast. Next, we study the affect of using the learned grasping model to

Algorithm 3 TrainGraspAutoCurr($G^1, ..., G^N$, $\mathcal{D}_g$, N)

if $|\mathcal{D}_g| \geq N_{max}$ **then**
  return TrainGrasp($G^1, ..., G^N$, $\mathcal{D}_g$, N, 0)
end if
$N_{since} = 0$
$r_{max} = 0$
for $n = 1, \ldots$ **do**
  Get grasp observation $\tilde{o}$.
  Sample $a_g \sim \pi_g(\cdot|\tilde{o})$. // see Equation 2
  Perform grasp a_g, receiving $r_g = 0$ or 1.
  Store $(\tilde{o}, a_g, r_g)$ in $\mathcal{D}_g$.
  if $|\mathcal{D}_g| \geq N_{bt}$ **then**
    Update $G^1, ..., G^N$ on $\mathcal{D}_g$
  end if
  if $r_g = 1$ **then**
    $N_{since} = 0$
    $r_{max} = 1$
  else
    $N_{since} = N_{since} + 1$
  end if
  if $|\mathcal{D}_g| \geq N_{max}$ **then**
    break
  end if
  if $r_{max} = 1$ **then**
    if $N_{since} \geq N_{stop}$ **then**
      break
    end if
  else
    if $N_{since} \geq N_{start}$ **then**:
      break
    end if
  end if
  if $r_g = 1$ **then**:
    Drop object randomly in grasping area.
  end if
end for
return r_{max}

relabel rewards for the navigation policy, as described in subsection 4.2. We find that using this relabeling method can increase learning speed and final policy quality, as shown in Figure 9 (left). Last, we compare the different curriculum methods, stationary and autonomous, in a simulated room with obstacles. The stationary curriculum method appears to results in faster learning but requires human intervention to train. The autonomous curriculum may find this environment difficult do to the obstacles which make exploratory navigation less successful.

Ablation. We compare our pseudo-reset method to a prior algorithm for reset-free learning [2], which learns how to perturb the environment between episodes of running the actual task policy by maximizing a novelty based reward. In Figure 4 (right) we see our simpler pseudo-reset is equally or more effective than the learned perturbation method, without requiring any additional learning machinery.

Algorithm 4 ReLMM with AutoCurr

Init: function estimators $\pi_n, G^1, \ldots, G^M$.
Replay buffers $\mathcal{D}_n = \{\}, \mathcal{D}_g = \{\}$
Deleted TrainGrasp($G^1, .., G^N$, D_g, N_{pt}, 1)
for $t = 0, \ldots, T$ steps **do**
  Get navigation observation o_t
  Sample $a_n \sim \pi_n(\cdot|o_t)$ and perform a_n
  if *uniform*() $\leq \mathbb{P}[\text{grasp}|o_t]$ **then**
    $r_g =$TrainGraspAutoCurr(G^1,.., G^M,D_g,N_{grasp})
  else $r_g = 0$
  Compute reward $r_n = r_g - 1$
  Get next navigation observation o_{t+1}
  Store (o_t, a_n, r_n, o_{t+1}) in $\mathcal{D}_n$.
  Update π_n with $\mathcal{D}_n$ using SAC.
  Pseudo-reset
end for

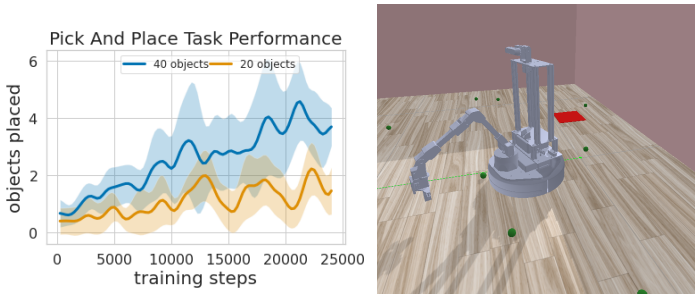

Figure 7: Left: Results for pick and place training, with either 40 or 20 objects available in the environment. We allow a maximum of 250 simulation steps for each evaluation rollout. Right: A snapshot from the proposed pick and place task with the red pace target behind the robot.

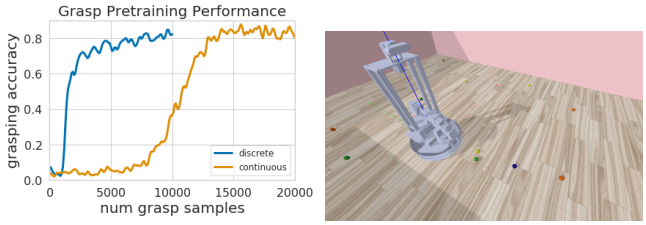

Figure 8: Left: Discrete grasping policies train significantly faster than continuous policies in our simulation ablation study. Right: Images from the simulated environment with diverse objects.

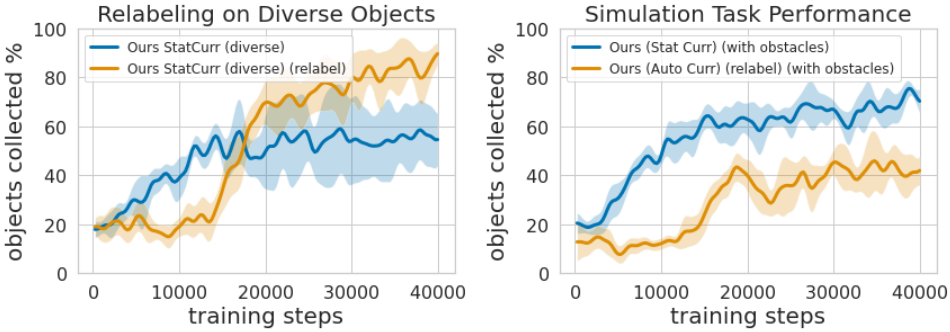

Figure 9: Further curriculum and relabeling comparison in simulation. We plot the performance for the simulated room with diverse objects (left) and with obstacles (right). We find that the relabeling reward helps significantly with the diverse objects because it encourages the navigation to go towards areas of high grasp uncertainty. However, the automatic curriculum in the obstacle room is still slower than the stationary curriculum, even with the best hyperparameters.