Exploration for Continually Improving Robots

Russell Mendonca

CMU-RI-TR-24-62

September 2024

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee

Deepak Pathak	Carnegie Mellon University (*Chair*)
Abhinav Gupta	Carnegie Mellon University
Ruslan Salakhutdinov	Carnegie Mellon University
Sergey Levine	University of California, Berkeley
Dorsa Sadigh	Stanford University

Thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics

Acknowledgements

This thesis is the culmination of a long journey in research enabled by immense support. I would like to thank my advisor Prof. Deepak Pathak for mentoring, guiding and supporting me. I have enjoyed our numerous (often late night) discussions on various research topics over the years. One of the most important things I learned from Deepak was how to ask ambitious research questions and set project goals, and to use desired functional capabilities to drive the investigation and design of approaches. Deepak also gave me ample freedom to explore different technical areas, allowing for a richer learning experience. I would also like to thank Prof. Abhinav Gupta, whose perspective on research problems has been very illuminating. I had a great experience interning at the Boston Dynamics AI Institute, and would like to especially thank my manager Jiugaung Wang for his support in helping me succeed and giving me a lot of latitude in trying different approaches.

My start in the area of reinforcement learning was enabled by an opportunity in my undergraduate days at UC Berkeley provided by Prof. Sergey Levine, for which I am extremely grateful. My early experience in his lab has continued to shape my thinking to this day, and I am very grateful for his support and encouragement, and in particular the mentorship of Prof. Abhishek Gupta (then a graduate student in the lab), that provided my first framework for approaching research.

I have learned a lot from interactions and discussions with various members of Deepak's lab, including Shikhar Bahl, Murtaza Dalal, Kenny Shaw, Alex Li, Ananye Agarwal, Lili Chen, Mihir Prabhudesai. I had the opportunity to collaborate with all of them on different projects, and now have the privilege of counting them among my friends. I'd also like to specifically thank Sudeep Dasari, Rohan Choudhury, Erica Weng, Suvansh Sanjeev, Laura Smith, Annie Xie, Michelle Zhao, Christina Baek, Gokul Swamy, Yasha Ektefaie, Joseph Baker, Anubha Kabra, Sharon John for their friendship and support over the years. These friendships, spanning across different locations and stages of my journey, have been a constant source of inspiration and support. Whether through the countless lunches, dinners and casual walks we enjoyed together or long-distance calls, occasional visits, and shared experiences during conferences, each of you has contributed uniquely to my growth and well-being. Your unwavering encouragement, insightful discussions, and genuine camaraderie have been invaluable, making this academic journey not just intellectually rewarding but also personally enriching. Thank you also for encouraging me to push my boundaries and try new things to enjoy life outside of work.

Finally, I would like to thank my parents, Celine and Nelson Mendonca. You were my very first mentors, and without your guidance, effort and conviction I would not be where I am today. Thank you for instilling in me the wonder and curiosity for learning new things and to continually challenge myself to grow. Your unconditional love and support has been invaluable throughout my journey.

Abstract

Data-driven learning is a powerful paradigm for enabling robots to learn skills. Current prominent approaches involve collecting large datasets of robot behavior via teleoperation or simulation, to then train policies. For these policies to generalize to diverse tasks and scenes, there is a large burden placed on constructing a rich initial dataset, which is bottle-necked by human labor required in collecting demonstrations or careful design of simulation assets and scenes. Can we instead enable robots to learn how to collect their own data for continual improvement? This thesis seeks to tackle this question of exploration, which directs how agents should act, leading to the discovery of useful behavior.

We first consider how to define exploration objectives even in the absence of rewards or demonstrations. To explore new goals, our key insight is that it is easier to identify action sequences that lead to some unknown goal state, than to generate the unknown goal directly. This is enabled by training a world model that can be used to measure the uncertainty of action sequences. For further efficiency for real world deployment, we decouple environment and agent-centric exploration. The former relates to incentivizing actions that lead to change in the visual features of objects which is often beneficial for manipulation tasks, and the latter to uncertainty of the robot's internal world model.

Next, we ask how to enable generalist robot explorers, for diverse tasks. Our approach is to learn data-driven priors to structure the action space, using human videos. We learn visual affordances, which characterize how objects can be interacted with by hands or end-effectors, providing a very efficient search space for exploration. Further this shared affordance action space can be used to train a joint human-robot world model. The model is first pre-trained on diverse video of human hands performing various tasks, and then fine-tuned with very few robot exploration trajectories. We also study how to efficiently adapt internet-scale video diffusion models using gradient information of a given reward function, which can enable future applications that use such models for planning in robotics.

The third question we consider is how to enable greater autonomy for robot explorers. We do so using mobile manipulation systems, as their extended feasible task space and resetting ability allows for continual practice and improvement with minimal human involvement. We show this on a quadruped equipped with an arm that learns to move chairs, sweep trash and vertically stand up dust-pans via real-world RL, as well as a custom wheeled system that learns to open doors across various buildings on campus. Finally, orthogonal to the question of exploration, we discuss how to scale data collection for bimanual dexterous manipulation using low-cost high-fidelity teleoperation as well as procedural scene generation in simulation to learn neural motion planners for robot arms. This is to obtain better initial policies from which robots can explore.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

A central goal in artificial intelligence is building agents that can sequentially make decisions to perform tasks. Robotics considers this problem in the most general setting, where the agent is embodied and can interact with the real world physically. We would like robots that can perform a diverse variety of tasks and adapt to perform new ones, in various environments and domains ranging from homes to factory settings to outer space. Recent advances in machine learning have shown amazing results in the fields of language and vision [40, 314, 327, 39, 397], by leveraging large internet-scale datasets and scaling up training compute. Inspired by this, there has been a renewed focus on data-driven learning for robotics. The current prevalent paradigm is to collect large datasets of demonstrations, and then use imitation learning [37, 36, 63, 458], or to leverage fast parallelizable simulators where agents learn specific tasks via reinforcement learning [206, 178, 152, 6, 144]. If we are to obtain general purpose policies using these approaches, there is a very large burden placed on collecting a representative set of expert demonstrations via teleoperation, or careful design of simulation environments and assets that capture various tasks, scenes and objects. These will need to be exhaustive enough to enable the resulting policy to *generalize* across various tasks and environments where robots might be deployed. However, it is very difficult to a priori anticipate all these settings.

In contrast, humans constantly learn new skills and improve in ability using their daily experience. For example, people frequently pick up new sports such as tennis, basketball or skiing, though most require quite a bit of practice before gaining proficiency. This critically depends on being able to interact with the world, and use new data to keep learning, instead of relying on capabilities that are frozen in time. Can we similarly enable agents that learn by collecting their *own interactive experience?* This leads to the question of exploration, which directs how agents should act to collect their own data in order to improve, which is the topic of this thesis.

How should agents direct their behavior to learn new skills and discover interesting states? In robot learning, policies are typically trained to maximize some notion of utility specified by a reward. Using this approach, separate reward functions have to be specified for each new skill that the robot has to learn, and the training process repeated. For example, using a reward for pushing objects, a robot will only take actions for that particular task, and will not seek out new skills such as picking or throwing. This is in stark contrast to how human children interact with the world, often engaging in seemingly mindless play that is not related to a specific task. However, this interaction endows them with rich, diverse experience which can be repurposed to accomplish a whole range of object manipulation behaviors. This is driven by what psychologists call 'intrinsic motivation' [334, 367]. Inspired by this, the first question we ask is how can we define **exploration objectives** for robots to exhibit similar behavior? We study objectives that quantify uncertainty of actions, and that incentivize changes in the visual features of objects.

These objectives allow the robot to discover skills, driven only by its interactions in its particular environment. Given a new environment, the process of discovery needs to be repeated. This leads us to ask if we can enable **generalist explorers**, which can utilize strong data-driven priors that are effective across environments. We look to human videos, since this is a rich data source depicting how to perform various tasks of interest, and is likely to keep increasing in size. Videos showing how humans interact with their environments communicate a top-down signal to robots of what are useful actions to undertake in a given scene. Importantly, we seek to extract actionable information from human videos in the form of 'affordances', which depict where and how to interact. Further, we show how these affordance action representations can be used to train *generalist world models*, effective for multiple tasks. This leads us to discuss the future of world models, likely to be enabled by recent advances in powerful video generation techniques [137, 39]. We investigate how to efficiently adapt generations from *pretrained internet-scale* video diffusion models, which can enable future applications to robotics.

While these data-driven priors enable robots to explore in a much more sample-efficient manner, continual improvement on tasks is bottleneck by the restricted autonomy of robots. Specifically, human intervention is often required to reset the scene, such as when objects are interacted with on a table and fall to the ground. Furthermore, many tasks of interest are precluded simply because they involve manipulation of larger, heavier objects, such as sweeping with a broom or opening a door. Returning to our example of learning for human children, we observe they engage in play mostly *autonomously*, without constant parent intervention. This is in large part due to mobility which greatly expands the set of reachable states. Hence, we next ask how to similarly enable better **autonomy** for robot exploration, using **mobile manipulation** systems? We show how to enable a Spot robot with an arm to learn multiple skills via practice and also build our own custom system for opening real-world doors across different buildings on campus.

So far we have discussed how robots can act to improve their performance. For robot deployment, an important orthogonal question is how competent the initial robot policy is. While exploration is important for improvement, for practical deployment there is no need to start with a randomly initialized policy. Hence, we also investigate approaches for **scaling robot data** critical for training initial deployment policies, both via low-cost teleoperation for bimanual dexterous tasks, and procedural scene generation in simulation for motion planning.

1.2 Contributions

The key questions this thesis studies are 1) How to define *exploration objectives*, even in the absence of rewards or demonstrations? 2) How to build *generalist robot explorers*, for diverse tasks? 3) How to enable greater *autonomy* for robot explorers, using *mobile manipulation* systems? Finally, we investigate approaches for scaling up data collection for robot learning, using low-cost teleoperation for imitation and procedural scene generation in simulation.

1.2.1 Exploration Objectives

The problem setting for these objectives involves *unsupervised exploration*, where the agent does not have access to any rewards or demonstrations. After a period of interaction with the environment, we test the agent with user-defined tasks.

Latent Explorer-Achiever: In Chapter 2, we present an algorithm that can discover skills and perform tasks in various simulation environments from raw image observations. This includes controlling a Franka robot to perform multi-stage tasks involving multiple objects such as a cabinet, light switch and kettle. The exploration objective involves having the agent setting goals for itself that are interesting. Prior work attempts this by sampling goals from the frontier of known states, using a measure of novelty [99, 285, 28, 295]. It is very difficult to directly generate goals that are far beyond the frontier of the agent's past experience, since this space is unknown. Our key insight is that it is instead easier to identify action sequences with uncertain outcomes. This is enabled by training a world model [140, 145, 146], that predicts outcomes of long sequences of actions in a learned latent space. We then use this to train a policy that prioritizes action sequences that have uncertain predictions, estimated via ensemble variance [287] in the world model latent space. These trajectories are likely to lead to states that fall beyond the agent's frontier of past experience, because otherwise the model would have exhibited low uncertainty. These discovered states are then used as goals to train a goal-reaching policy in the world model's imagination, which enables the agent to perform user-defined tasks.

Environment and Agent Centric Objectives: Real robot deployment requires an exploration algorithm with greater sample efficiency. We enable this in Chapter 3 by decoupling *environment-centric* and *agent-centric* exploration. The former relates to changes in visual features of objects in the world (computed using the segmentation mask of the robot), and is independent of learning progress made by the agent, while the latter corresponds to the agent's internal model of the world. Robots that prioritize actions resulting in changes in the environment will encounter richer contact interactions, which is very useful for acquiring manipulation skills. However, just relying on this environment-centric signal is not sufficient, as it is constant for a given state-action pair and does not evolve with the agent's behavior. For agent-centric exploration, we encourage the robot to be curious *about the environment change*, using the approach described in Chapter 2. This will cause it to try to interact with objects in different ways, preventing stagnation of robot exploration behavior. Using this approach, a Franka robot learns to perform different tasks such as picking up a knife, opening shelves and cabinets with only around 100 trajectories in 1-2 hours in two distinct play kitchens, without any human rewards.

1.2.2 Generalist Explorers using Prior Video

We extract affordances from human videos for deployment on robots, and investigate using the shared action space for training joint human-robot world models. We also study efficiently adapting generations from pretrained video diffusion models.

Affordances from Human Videos: The utility of human video data has been recognized in the robot learning community, and there have been recent efforts to endow robots with pretrained visual representations [355, 273, 241, 429, 309, 436]. However, since the state space complexity grows exponentially with actions, we wanted to extract *actionable* representations from human video for robot control. In Chapter 4, we approached this by learning visual affordances [20], which characterize ways in which objects can be interacted with. We use a robotics-first approach, extracting *what* to focus on in a scene, and *how* to interact with it. This is represented by a heat-map which indicates regions in the scene where the human hand likely makes contact, and key-points for the subsequent post-contact trajectory. This provides a very efficient action space for exploration, for the robot end-effector to operate in. The robot collects real-world samples in this space to refine its uncertainty estimate of the contact and post-contact directions, based on task reward. We extensively evaluate this affordance representation on a Franka arm and a Stretch robot across 8 different manipulation tasks, for different robot-learning paradigms including data collection for imitation, goal-reaching and unsupervised exploration.

Joint Human-Robot World Models: To move towards generalist robots, we wanted to build a system that can leverage the commonality between different tasks. In Chapter 5, we trained a single *joint* world model, which is first pretrained on diverse video of human hands performing various tasks, and then fine-tuned with very few robot trajectories. The key idea that enables this shared world model is to use the common affordance action space, which we developed in Chapter 4. Defining this common shared action space allows the model to capture how hands and robot end-effectors grasp and interact with various objects. The interaction trajectories for fine-tuning do not require any task supervision and can be obtained simply by exploring in the visual affordance space. We note that both pretraining on human videos and fine-tuning the world model on robot data do not make any assumption on rewards, and this *unsupervised* setting allows us to utilize data relevant for different tasks. Hence, we can train a *single* world model on all the data, thus enabling generalist agents. After fine-tuning, the world model can be used to perform specific robot tasks indicated via goal images simply by planning through the model, using image feature distance as reward.

Video Diffusion Alignment: Given the utility of world models for enabling generalists, we study likely future implementations that leverage recent advances. This includes internet-scale video diffusion models, that can generate future image frames, capture scene and concept generalization, and aspects of interaction physics between different complex objects [39, 137]. One way for such models to be used for robotics is to efficiently adapt video generation to depict how to reach a particular desired goal configuration. The predominant approach for this for language and image generation has been supervised fine-tuning [312, 38], but the required dataset collection is much more costly for videos. Furthermore, the process needs to be repeated for every new video generation task. In Chapter 6, we use reward models evaluated on the video generations, specifically by backpropogating gradients. This allows the model to be updated very efficiently in terms of compute and the number of queries needed to the reward model. In language and image domains, utilization of reward models is mostly gradient-free [31, 219], but we find the efficiency gap becomes very large in the video setting, making it crucial to use reward gradients.

1.2.3 Autonomous Explorers with Mobile Manipulators

Mobile manipulation systems enable greater autonomy for robot exploration due to their extended reach, and larger set of resettable states. Furthermore, they enable attempting tasks involving larger or heavier objects, as well as those outside the lab.

Autonomous Spot RL: In Chapter 7, we present a general approach for continuously improving mobile manipulation skills directly in the real world with autonomous RL, with minimal human intervention. We enable higher-quality data collection by guiding exploration toward object interactions using off-the-shelf visual models. This leads the robot to search for, navigate to, and grasp objects before learning how to manipulate them. We preserve state diversity to prevent robot stagnation by extending the approach of goal-cycles to mobile tasks and with multi-robot systems. For sample efficient policy learning, we combine RL with *behavior priors* that contain basic task knowledge. These priors can be planners with a simplified incomplete model, or procedurally generated motions. For rewards without instrumentation or human involvement, we combine semantic information from detection and segmentation models with low-level depth observations for object state estimation. Our approach enables a 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 goal region.

Open-World Door Opening: In Chapter 8, we present a full-stack approach for taking robots out of the lab to manipulate real-world doors, cabinets, drawers, and refrigerators in open-ended unstructured environments, building our own low-cost mobile manipulation platform. In order to effectively manipulate objects in open-world settings, we use an *adaptive learning* approach, where the robot keeps learning from online samples collected during interaction. Consider the manner in which people typically approach operating articulated objects such as doors. This generally first involves reaching towards a part of the object, such as a handle, and establishing a grasp. We then execute constrained manipulation like rotating, unlatching, or unhooking, where we apply arm or body movement to manipulate the object. In addition to this high-level strategy, there are also lower-level decisions made at each step regarding exact direction of movement, extent of perturbation and amount of force applied. Inspired by this, we use a hierarchical action space for our controller, where the high-level action sequence follows the grasp, constrained manipulation strategy. These primitives are parameterized by learned low-level continuous values, which needs to be adapted to operate diverse articulated objects. To further bias the exploration of the system towards reasonable actions during online sampling, we collect a dataset of expert demonstrations on 12 training objects, including doors, drawers and cabinets to train an initial policy via behavior cloning.

1.2.4 Scaling Robot Learning

Orthogonal to the question of exploration, we study approaches for obtaining competent initial policies, via low-cost teleoperation for very high-dimensional bimanual dexterous systems, and procedural scene generation in simulation to train policies for motion planning. This brings us closer to robot systems that can be deployed.

Bimanual Dexterous Teleoperation: General purpose robots will need to operate in environments built around humans, and execute tasks that correspond to activities that people can perform. Hence, one approach to building such systems is to use a hardware form factor that resembles humans with two arms each equipped with a dexterous multi-fingered hand. In Chapter 9, we present BiDex, a system for dexterous low-cost teleoperation for bimanual hands and arms, in-the-wild, in any environment. Wu et al. [426] show that a low-cost 3D printed scaled teacher arm model that has the same kinematic link structure of a large robot arm can be used for accurate and effective arm tracking, by obtaining supervision in joint-angle space. However, this only provides one degree for freedom (DOF) for finger tracking instead of the twenty two plus DOF of the human hand. Meanwhile, wearable gloves have been increasingly used in recent years [408, 362, 246] for hand tracking which record the fingertip position of the human hand using EMF sensors. Our key insight is to combine kinematic link structure for arm tracking along with a motion capture fingertip glove for tracking finger movements accurately and reliably. The data collected by the system can be used to train effective policies using imitation learning, after which the system can perform the task autonomously. We compare the accuracy and speed of data collection to virtual reality based teleoperation systems.

Neural Motion Planning: Previous approaches to motion planning [215, 33, 185, 318, 346, 154, 226, 200] are often slow at producing solutions since they largely plan from scratch at test time, re-using little to no information outside of the current problem and what is engineered by a human designer. In Chapter 10, we show that large scale data generation in simulation can enable training generalist policies that can be successfully deployed for real-world motion planning tasks. We build a large number of complex environments by combining procedural, programmatic assets with models of everyday objects sampled from large 3D datasets. These are used to collect expert data from state-of-the-art (SOTA) motion planners [380], which we then *distill* into a reactive, generalist policy. Since this policy has seen data from 1 million scenes, it is capable of generalizing to novel obstacles and scene configurations that it has never seen before, including variation across poses, objects, obstacles, backgrounds, scene arrangements, in-hand objects, and start/goal pairs.

1.3 Publications

List of publications included in this thesis:

Chapter 2: Discovering and Achieving Goals via World Models. NeurIPS 2021 [250].

Chapter 3: Autonomously Exploring Robotic Agents in the Real World. ICRA 2023 [251].

Chapter 4: Affordances from Human Videos as a Versatile Representation for Robotics. CVPR 2023 [20].

Chapter 5: Structured World Models from Human Videos. RSS 2023 [252].

Chapter 6: Video Diffusion Alignment via Reward Gradients. In Submission [298].

Chapter 7: Continuously Improving Mobile Manipulation with Autonomous Real-World RL. CoRL 2024 [253].

Chapter 8: Adaptive Mobile Manipulation for Articulated Objects In the Open World. In Submission [432].

Chapter 9: Bimanual Dexterity for Complex Tasks. CoRL 2024 [363].

Chapter 10: Neural MP: A Generalist Neural Motion Planner. In Submission [75].

Part I

Exploration Objectives

Chapter 2

Discovering and Achieving Goals via World Models

2.1 Motivation

How can we build an agent that learns to solve hundreds of tasks in complex visual environments, such as rearranging objects with a robot arm or completing chores in a kitchen? While traditional reinforcement learning (RL) has been successful for individual tasks, it requires a substantial amount of human effort for every new task. Specifying task rewards requires domain knowledge, access to object positions, is time-consuming, and prone to human errors. Moreover, traditional RL would require environment interaction to explore and practice in the environment for every new task. Instead, we approach learning hundreds of tasks through the paradigm of unsupervised goal-conditioned RL, where the agent learns many diverse skills in the environment in the complete absence of supervision, to later solve tasks via user-specified goal images immediately without further training [181, 341, 8].

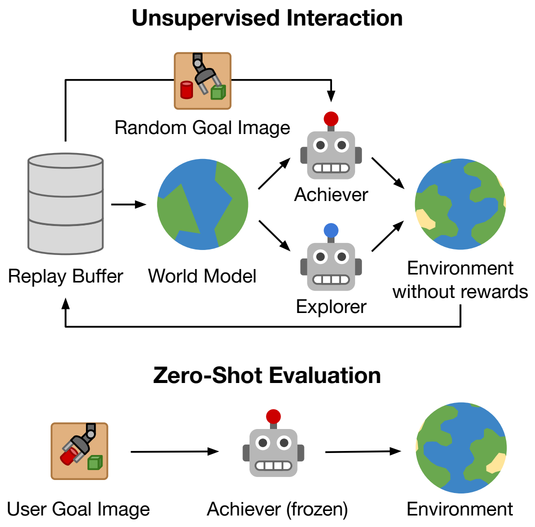


Figure 2.1: LEXA learns a world model without any supervision, and leverages it to train two policies in imagination. The *explorer* finds new images and the *achiever* learns to reliably reach them. Once trained, the achiever reaches user-specified goals zero-shot without further training at test time.

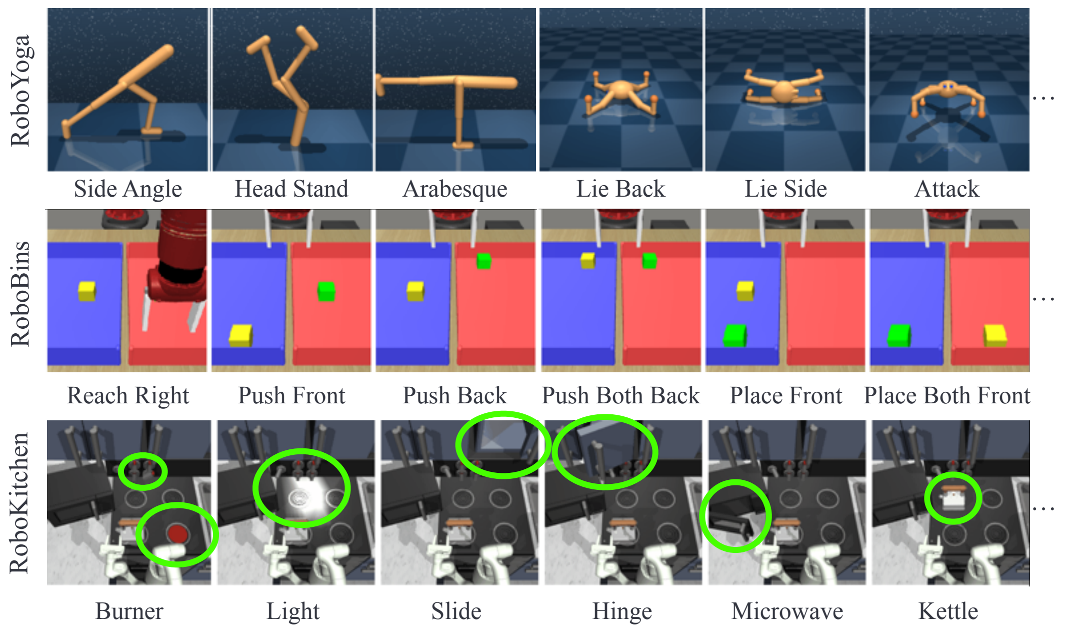


Figure 2.2: We benchmark LEXA across four visual control environments. A representative sample of the test-time goals is shown here. RoboYoga features complex locomotion and precise control of high-dimensional agents, RoboBins manipulation with multiple objects, and RoboKitchen a variety of diverse tasks that require complex control strategies such as opening a cabinet.

Challenges: Exploring the environment and learning to solve many different tasks is substantially more challenging than traditional RL with a dense reward function or learning from expert demonstrations. Existing methods are limited to simple tasks, such as picking or pushing a puck [97, 271, 295] or controlling simple 2D robots [414]. The key challenge in improving the performance of unsupervised RL is *exploration*. In particular, previous approaches explore by either revisiting previously seen rare goals [109, 99, 457] or sampling goals from a generative model [271, 295]. However, in both these approaches, the policy as well as the generative model are trained on previously visited states from the replay buffer, and hence the sampled goals are either within or near the frontier of agent's experience. Ideally, we would like the agent to discover goals much beyond its frontier for efficient exploration, but how does an agent generate goals that it is yet to encounter? This is an open question not just for AI but for cognitive science too [348].

Approach: To rectify this issue, we leverage a learned world model to train a separate *explorer* and *achiever* policy in imagination. Instead of randomly sampling or generating goals, our explorer policy discovers distant goals by first planning a sequence of actions optimized in imagination of the world model to find novel states with high expected information gain [230, 366, 349]. It then executes those imagined actions in the environment to discover interesting states without the need to generate them. Note these actions are likely to lead the agent to states which are several steps outside the frontier because otherwise the model wouldn't have had high uncertainty or information gain. Finally, these discovered states are used as diverse targets for the achiever to practice. We train the achiever from on-policy imagination rollouts within the world model and without relying on experience relabeling, therefore leveraging *foresight over hindsight*. After this unsupervised training phase, the achiever solves tasks specified as goal images zero-shot without any additional learning at deployment. Unlike in the conventional RL paradigm [258, 386], our method is trained once and then used to achieve several tasks at test time without any supervision during training or testing.

Contributions: We introduce Latent Explorer Achiever (LEXA), an unsupervised goal reaching agent that trains an explorer and an achiever within a shared world model. At training, LEXA unlocks diverse data for goal reaching in environments where exploration is nontrivial. At test time, the achiever solves challenging locomotion and manipulation tasks provided as user-specified goal images. Our contributions are summarized as follows:

- We propose to learn separate explorer and achiever policies as an approach to overcome the exploration problem of unsupervised goal-conditioned RL.
- We show that forward-looking exploration by planning with a learned world model substantially outperforms previous strategies for goal exploration.
- To evaluate on challenging tasks, we introduce a new goal reaching benchmark with a total of 40 diverse goal images across 4 different robot locomotion and manipulation environments.
- LEXA outperforms prior methods, being the first to show success in the Kitchen robotic manipulation environment, and achieves goal images where multiple objects need to be moved.

2.2 Latent Explorer Achiever (LEXA)

Our aim is to build an agent that can achieve arbitrary user-specified goals after learning in the environment without any supervision. This presents two challenges - collecting trajectories that contain diverse goals and learning to achieve these goals when specified as a goal image. We introduce a simple solution based on a world model and imagination training that addresses both challenges. The world model represents the agent's current knowledge about the environment and is used for training two policies, the explorer and the achiever. To explore novel situations, we construct an estimate of which states the world model is still uncertain about. To achieve goals, we train the goal-conditioned achiever in imagination, using the images found so far as unsupervised goals. At test time, the achiever is deployed to reach user-specified goals. The training procedure is in Algorithm 1.

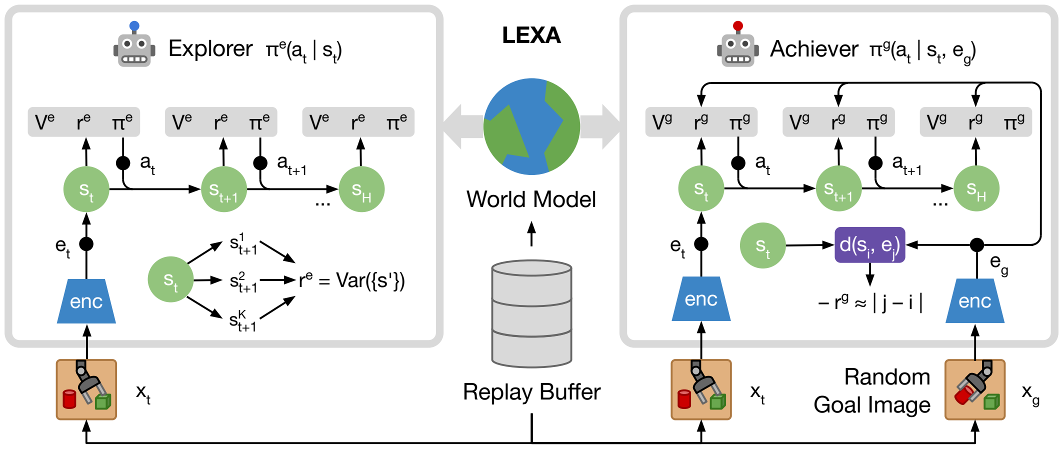


Figure 2.3: Latent Explorer Achiever (LEXA) learns a general world model that is used to train an explorer and a goal achiever policy. The explorer (left) is trained on imagined latent state rollouts of the world model $s_{t:T}$ to maximize the disagreement objective $r_t^e = \mathrm{Var}(s')$. The goal achiever (right) is conditioned on a goal g and is also trained on imagined rollouts to minimize a distance function $d(s_t, e_g)$. Goals are sampled randomly from replay buffer images. For training a temporal distance, we use the imagined rollouts of the achiever and predict the number of time steps between each two states. By combining forward-looking exploration and data-efficient training of the achiever, LEXA provides a simple and powerful solution for unsupervised reinforcement learning.

2.2.1 World Model

To efficiently predict potential outcomes of future actions in environments with high-dimensional image inputs, we leverage a Recurrent State Space Model (RSSM) [145] that learns to predict forward using compact model states that facilitate planning [417, 43]. In contrast to predicting forward in image space, the model states enable efficient parallel planning with a large batch size and can reduce accumulating errors [340]. The world model consists of the following components:

$$\begin{array}{llll} \text{Encoder:} & e_t = \mathrm{enc}_\phi(x_t) & \text{Posterior:} & q_\phi(s_t \mid s_{t-1}, a_{t-1}, e_t) \\ \text{Dynamics:} & p_\phi(s_t \mid s_{t-1}, a_{t-1}) & \text{Image decoder:} & p_\phi(x_t \mid s_t) \end{array}$$

The model states s_t contain a deterministic component h_t and a stochastic component z_t with diagonal-covariance Gaussian distribution. h_t is the recurrent state of a Gated Recurrent Unit (GRU) [66]. The encoder and decoder are convolutional neural networks (CNNs) and the remaining components are MLPs. The world model is trained end-to-end by optimizing the evidence lower bound (ELBO) via stochastic backpropagation [196, 323] with the Adam optimizer [195].

Algorithm 1 Latent Explorer Achiever (LEXA)

```
1: initialize: World model $\mathcal{M}$, Replay buffer $\mathcal{D}$,
2: Explorer $\pi^{\mathrm{e}}(a_t \mid z_t)$, Achiever $\pi^{\mathrm{g}}(a_t \mid z_t, g)$
3: while exploring do
4:     Train $\mathcal{M}$ on $\mathcal{D}$
5:     Train $\pi^{\mathrm{e}}$ in imagination of $\mathcal{M}$ to maximize exploration rewards $\sum_t r_t^{\mathrm{e}}$
6:     Train $\pi^{\mathrm{g}}$ in imagination of $\mathcal{M}$ to maximize $\sum_t r_t^{\mathrm{g}}(z_t, g)$ for images $g \sim \mathcal{D}$.
7:     (Optional) Train $d(z_i, z_j)$ to predict distances $j - i$ on the imagination data
   from last step.
8:     Deploy $\pi^{\mathrm{e}}$ in the environment to explore and grow $\mathcal{D}$.
9:     Deploy $\pi^{\mathrm{g}}$ in the environment to achieve a goal image $g \sim \mathcal{D}$ to grow $\mathcal{D}$.
10: end while
11: while evaluating do
12:     given: Evaluation goal $g$
13:     Deploy $\pi^{\mathrm{g}}$ in the world to reach $g$.
14: end while
```

2.2.2 Explorer

To efficiently explore, we seek out surprising states imagined by the world model [343, 384, 366, 349, 42], as opposed to retrospectively exploring by revisiting previously novel states [28, 285, 44, 30]. As the world model can predict model states that correspond to unseen situations in the environment, the imagined trajectories contain more novel goals, compared to model-free exploration that is limited to the replay buffer. To collect informative novel trajectories in the environment, we train an exploration policy π^e from the model states s_t in imagination of the world model to maximize an exploration reward:

$$\text{Explorer:} \qquad \pi^e(a_t \mid s_t) \qquad \text{Explorer Value:} \qquad v^e(s_t)$$

To explore the most informative model states, we estimate the epistemic uncertainty as a disagreement of an ensemble of transition functions. We train an ensemble of 1-step models to predict the next model state from the current model state. The ensemble model is trained alongside the world model on model states produced by the encoder q_ϕ. Because the ensemble models are initialized at random, they will differ, especially for inputs that they have not been trained on [210, 287]:

$$\text{Ensemble:} \quad f(s_t, \theta^k) = \hat{z}^k_{t+1} \quad \text{for} \quad k = 1..K$$

Leveraging the ensemble, we estimate the epistemic uncertainty as the ensemble disagreement. The exploration reward is the variance of the ensemble predictions averaged across dimension of the model state, which approximates the expected information gain [22, 349]:

$$r_t^{\mathrm{e}}(s_t) \doteq \frac{1}{N} \sum_n \mathrm{Var}_{\{\mathrm{k}\}} \left[f(s_t, \theta_k) \right]_n$$

The explorer π^e maximizes the sum of future exploration rewards r_t^e using the Dreamer algorithm [146], which considers long-term rewards into the future by maximizing λ-returns under a learned value function. As a result, the explorer is trained to seek out situations are as informative as possible from imagined latent trajectories of the world model, and is periodically deployed in the environment to add novel trajectories to the replay buffer, so the world model and goal achiever policy can improve.

2.2.3 Achiever

To leverage the knowledge obtained by exploration for learning to reach goals, we train a goal achiever policy π^g that receives a model state and a goal as input. Our aim is to train a general policy that is capable of reaching many diverse goals. To achieve this in a data-efficient way, it is crucial that environment trajectories that were collected with one goal in mind are reused to also learn how to reach other goals. While prior work addressed this by goal relabeling which makes off-policy policy optimization a necessity [8], we instead leverage past trajectories via the world model trained on them that lets us generate an unlimited amount of new imagined trajectories for training the goal achiever on-policy in imagination. This simplifies policy optimization and can improve stability, while still sharing all collected experience across many goals.

$$\text{Achiever:} \qquad \pi^g(a_t \mid s_t, e_g) \qquad \text{Achiever Value:} \qquad v^g(s_t, e_g)$$

To train the goal achiever, we sample a goal image x_g from the replay buffer and compute its embedding $e_g = \mathrm{enc}_\phi(x_g)$. The achiever aims to maximize an unsupervised goal-reaching reward $r^g(s_t, e_g)$. We discuss different choices for this reward in Sec. 2.2.4. We again use the Dreamer algorithm [146] for training, where now the value function also receives the goal embedding as input.

In addition to imagination training, it can also be important to perform practice trials with the goal achiever in the true environment, so that any model inaccuracies along the goal reaching trajectories may be corrected. To perform practice trials, we sample a goal from the replay buffer and execute the goal achiever policy for that goal in the environment. These trials are interleaved with exploration episodes collected by the exploration policy in equal proportion. We note that the goal achiever learning is entirely unsupervised because the practice goals are simply images the agent encountered through exploration or during previous practice trails.

2.2.4 Latent Distances

Training the achiever policy requires us to define a goal achievement reward $r^g(s_t, e_g)$ that measures how close the latent state s_t should be considered to the goal e_g. One simple measure is the cosine distance in the latent space obtained by inputting image observations into the world-model. However, such a distance function brings *visually* similar states together even if they could be farther apart in *temporal* manner as measured by actions needed to reach from one to other. This bias makes this suitable only to scenarios where most of pixels in the observations are directly controllable, e.g., trying to arrange robot's body in certain shape, such as RoboYoga poses in Figure 2.2. However, many environments contain agent as well as the world, such as manipulation involves interacting with objects that are not directly controllable. The cosine distance would try matching the entire goal image, and thus places a large weight on both matching the robot and object positions with the desired goal. Since the robot position is directly controllable it is much easier to match, but this metric overly focuses on it, yielding poor policies that ignore objects. We address this is by using the number of timesteps it takes to move from one image to another as a distance measure [181, 155]. This ignores large changes in robot position, since these can be completed in very few steps, and will instead focus more on the objects. This temporal cost function can be learned purely in imagination rollouts from our world model allowing as much data as needed without taking any steps in the real world.

Cosine Distance: To use cosine distance with LEXA, for a latent state s_t, and a goal embedding e^g, we use the latent inference network q to infer s^g, and define the reward as the cosine similarity [455]:

$$r_t^g(s_t, e_g) \doteq \sum_i \overline{s}_{ti}\overline{s}_{gi}, \quad \text{where} \quad \overline{s}_t = s_t/\|s_t\|_2, \quad \overline{s}_g = s_g/\|s_g\|_2,$$

i.e. the cosine of the angle between the two vectors s_t, s_g in the $N-$dimensional latent space.

Temporal Distance: To use temporal distances with LEXA, we train a neural network d to predict the number of time steps between two embeddings. We train it by sampling pairs of states s_t, s_{t+k} from an imagined rollout of the achiever and predicting the distance k. We implement the temporal distance in terms of predicted image embeddings $\hat{e}_{t+k}$ in order to remove extra recurrent information:

$$\text{Predicted embedding:} \quad \text{emb}(s_t) = \hat{e}_t \approx e_t \quad \text{Temporal distance:} \quad d_\omega(\hat{e}_t, \hat{e}_{t+k}) \approx k/H,$$

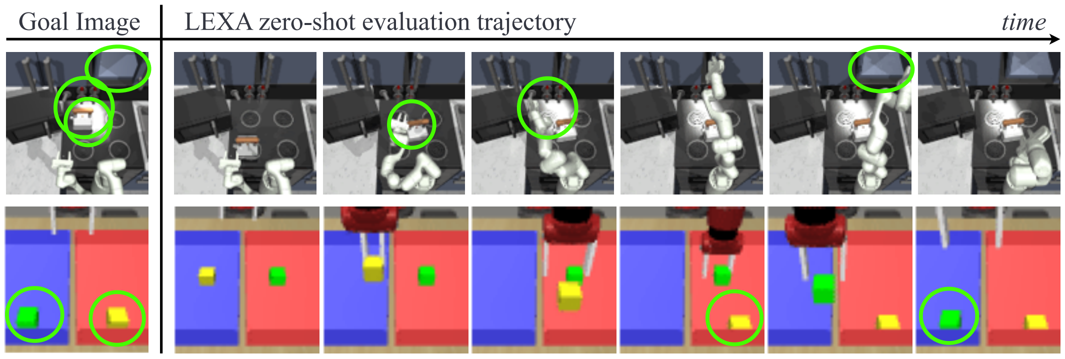


Figure 2.4: Successful LEXA trajectories. When given a goal image from the test set, LEXA's achiever is used in the environment to reach that image. On RoboKitchen, LEXA manipulates up to three different objects together from a single goal image (kettle, light switch, and cabinet). On RoboBins, LEXA performs temporally extended tasks such as picking and placing two objects in a row.

where H is the maximum distance equal to the imagination horizon. Training distance function only on imagination data from the same trajectory would cause it to predict poor distance to far away states coming from other trajectories, such as images that are impossible to reach during one episode. In order to incorporate learning signal from such far-away goals, we include them by sampling images from a different trajectory. We annotate these negative samples with the maximum possible distance, so the agent prefers images that were seen in the same trajectory.

$$r_t^g(s_t, e_g) = -d_\omega(\hat{e}_t, e_g), \quad \text{where} \quad \hat{e}_t = \text{emb}(s_t), \quad e_g = \text{enc}_\phi(x_g)$$

The learned distance function depends on the training data policy. However, as the policy becomes more competent, the distance estimates will be closer to the optimal number of time steps to reach a particular goal, and the policy converges to the optimal solution [155]. LEXA always uses the latest data to train the distance function using imagination, ensuring that the convergence is fast.

2.3 Experiments

Our evaluation focuses on the following scientific questions:

1. Does LEXA outperform prior work on previous benchmarks and a new challenging benchmark?
2. How does forward-looking exploration of goals compare to previous goal exploration strategies?

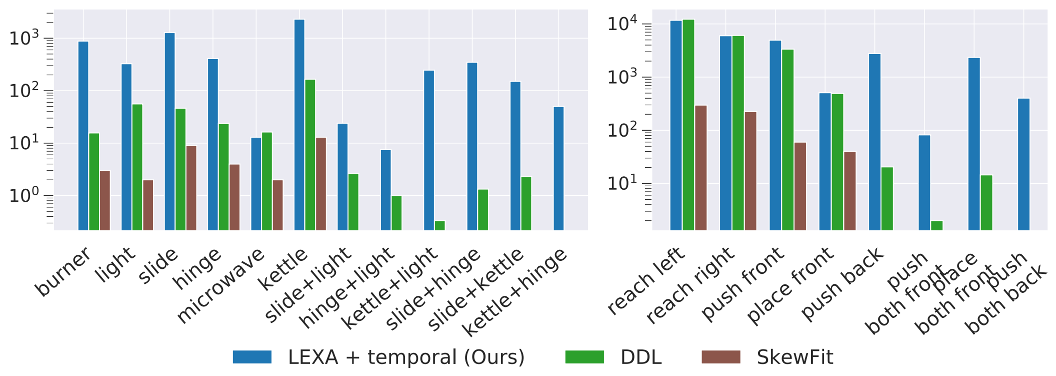


Figure 2.5: Coincidental goal success achieved during the unsupervised exploration phase. The forward-looking explorer policy of LEXA results in substantially better coverage compared to SkewFit, a popular method for goal based exploration.

3. How does the distance function affect the ability to reach goals in different types of environments?
4. Can we train one general LEXA to control different robots across visually distinct environments?
5. What components of LEXA are important for performance?

We evaluate LEXA on prior benchmarks used by SkewFit [295], DISCERN [414], and Plan2Explore [349] in Sec. 2.3.3. Since these benchmarks are largely saturated, we also introduce a new challenging benchmark shown in Figure 2.2. We evaluate LEXA on this benchmark is Sec. 2.3.2.

2.3.1 Experimental setup

As not many prior methods have shown success on reaching diverse goals from image inputs, we perform an apples-to-apples comparison by implementing the baselines using the same world model and policy optimization as our method.

- **SkewFit:** SkewFit [295] uses model-free hindsight experience replay and explores by sampling goals from the latent space of a variational autoencoder [196, 323]. Being one of the state-of-the-art agents, we use the original implementation that does not use a world model or explorer policy.
- **DDL:** Dynamic Distance Learning [155] trains a temporal distance function similar to our method. Following the original algorithm, DDL uses greedy exploration and trains the distance function on the replay buffer instead of in imagination.

- **DIAYN:** Diversity is All You Need [102] learns a latent skill space and uses mutual information between skills and reached states as the objective. We augment DIAYN with our explorer policy and train a learned skill predictor to obtain a skill for a given test image [67].

- **GCSL:** Goal-Conditioned Supervised Learning [123] trains the goal policy on replay buffer goals and mimics the actions that previously led to the goal. We also augment GCSL with our explorer policy, as we found no learning success without it.

Our new benchmark defines goal images for a diverse set of four existing environments as follows:

- **RoboYoga:** We use the walker and quadruped domains of the DeepMind Control Suite [391] to define the RoboYoga benchmark, consisting of 12 goal images that correspond to different body poses for each of the two environments, such as lying down, standing up, and balancing.

- **RoboBins:** Based on MetaWorld [448], we create a scene with a Sawyer robotic arm, two bins, and two blocks of different colors. The goal images specify tasks that include reaching, manipulating only one block, and manipulating both blocks.

- **RoboKitchen:** The last benchmark involves the challenging kitchen environment from [134], where a franka robot can interact with various objects including a burner, light switch, sliding cabinet, hinge cabinet, microwave, or kettle. The goal images we include describe tasks that require interacting with only one object, as well as interacting with two objects.

Method	**Kitchen**	**RoboBins**	**Quadruped**	**Walker**
DDL	0.00	35.42	22.50	40.00
DIAYN	0.00	13.69	13.81	0.28
GCSL	0.00	7.94	15.83	1.11
SkewFit	0.23	15.77	5.52	0.01
LEXA + Temporal (Ours)	**37.50**	**69.44**	31.39	36.72
LEXA + Cosine (Ours)	6.02	45.83	**56.11**	**73.06**

Table 2.1: Performance on our new challenging benchmark, spanning across the four domains shown in Figure 2.2. The number are goal success rates, averaged over test goals within each environment.

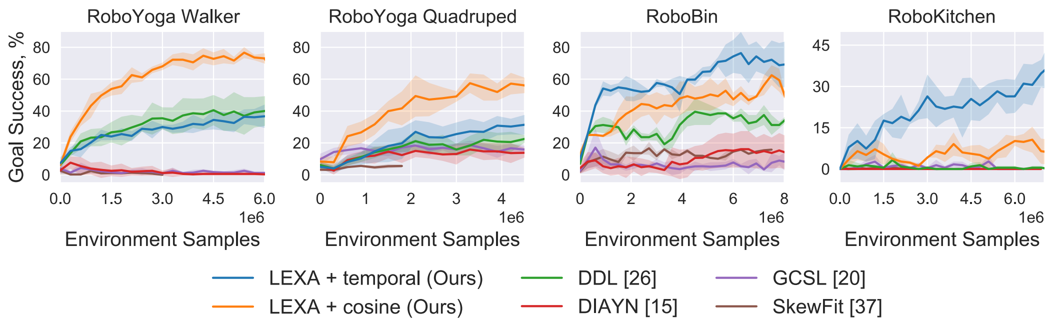


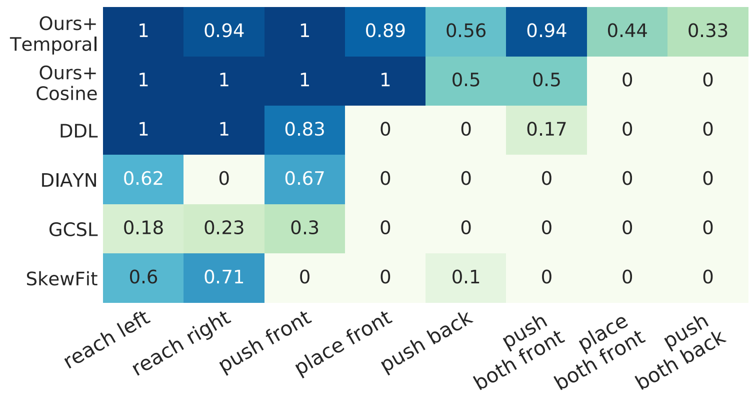
Figure 2.6: Evaluation of goal reaching agents on our four benchmarks. A single agent is trained from images without rewards and then evaluated on reaching goal images from the test set (see Figure 2.1). Both LEXA agents solve many of the tasks and significantly outperform prior work. SkewFit and DLL struggle with exploration, while DIAYN and GCSL use our explorer but still are not able to learn a good downstream policy. Refer table 2.1 for final success percentage (averaged across tasks) for each method and benchmark domain.

2.3.2 Performance on New Benchmark

We show the results on our main benchmark in Figure 2.6 and include heatmaps that show per-task success on each of the evaluation tasks from the benchmarks in the Appendix. Further, we report success averaged across tasks for each domain at the end of training in Table 2.1. We visualize example successful trajectory executions for tasks that require manipulating multiple objects in Fig. 2.4.

RoboYoga: The environments in this benchmark are directly controllable since they contain no other objects except the robot. We recall that for such settings we expect the cosine distance to be effective, as perceptual distance is quite accurate. Training is thus faster compared to using learned temporal distances, where the metric is learned from scratch. From Table 2.1 and Figure 2.6 we see that this is indeed the case for these environments (Walker and Quadruped), as LEXA with the cosine metric outperforms all prior approaches. Furthermore with temporal distances LEXA makes better progress compared to prior work on a much larger number of goals as can be seen from the per-task performance (Figures L.2, L.3), even though average success over goals looks similar to that of DDL.

RoboBins: This environment involves interaction with block objects, and thus is not directly controllable, and so we expect LEXA to perform better with the temporal distance metric. From Table 2.1 and Figure 2.6, we see that LEXA gets higher average success than all prior approaches. Further from the per-task performance in 2.7, LEXA with the temporal distance metric is the only approach that makes progress on all goals in the benchmark. The main difference in performance between using



	reach left	reach right	push front	place front	push back	push both front	place both front	push both back
Ours+ Temporal	1	0.94	1	0.89	0.56	0.94	0.44	0.33
Ours+ Cosine	1	1	1	1	0.5	0.5	0	0
DDL	1	1	0.83	0	0	0.17	0	0
DIAYN	0.62	0	0.67	0	0	0	0	0
GCSL	0.18	0.23	0.3	0	0	0	0	0
SkewFit	0.6	0.71	0	0	0.1	0	0	0

Figure 2.7: Success rates on RoboBin. In line with the prior literature, previous methods are successful at reaching and sometimes pushing. LEXA pushes the state-of-the-art by picking and placing multiple objects to reach challenging goal images. Analogous heat maps for the other domains are included in the appendix.

temporal and cosine distance can be seen in the tasks involving two blocks, which are the most complex tasks in this environment (the last 3 columns of the per-task plot). The best performing prior method is DDL which solves reaching, and can perform simple pushing tasks. This method performs poorly due to poor exploration, as shown in Figure 2.5. We see that while other prior methods make some progress on reaching, they fail on harder tasks.

RoboKitchen: This benchmark involves diverse objects that require different manipulation behavior. From Tab. 2.1 and Figure 2.6 we find that LEXA with temporal distance is able to learn multiple RoboKitchen tasks, some of which require sequentially completing 2 tasks in the environment. All prior methods barely make progress due to the challenging nature of this benchmark, and furthermore using the cosine distance function makes very limited progress. The gap in performance between using the two distance functions is much larger in this environment compared to RoboBins since there are many more objects not as clearly visible as the blocks.

Single Agent Across All Environments: In the previous sections we have shown that our approach can achieve diverse goals in different environments. However, we trained a new agent for every new environment, which doesn't scale well to large numbers of environments. Thus we investigate if we can train a train a single agent across four environments in the benchmark. From Figure L.6 we see that our approach with learned temporal distance is able to make progress on tasks from RoboKitchen, RoboBins Reaching, RoboBins Pick & Place and Walker, while the best prior method on the single-environment tasks (DDL) mainly solves walker tasks and reaching from RoboBin.

2.3.3 Performance on Prior benchmarks

To further verify the results obtained on our benchmark, we evaluate LEXA on previously used benchmarks. We observe that LEXA significantly outperforms prior work on these benchmarks, and is often close to the optimal policy.

Table 2.2: Goal distance for SkewFit goals [295].

Method	Pusher	Pickup
RIG [271]	7.7cm	3.7cm
RIG + HER [8]	7.5cm	3.5cm
Skew-Fit [295]	4.9cm	1.8cm
LEXA + Temporal	**2.3cm**	**1.4cm**

SkewFit Benchmark: SkewFit [295] introduces a robotic manipulation benchmark for unsupervised methods with simple tasks like planar pushing or picking. We evaluate on this benchmark in Tab. 2.2. Baseline results are taken from [295]. LEXA significantly outperforms prior work on these tasks.

DISCERN Benchmark: We attempted to replicate the tasks described in [414] that are based on simple two-dimensional robots [391]. While the original tasks are not released, we followed the procedure for generating the goals described in the paper. Despite following the exact procedure, we were not able to obtain similar goals to the ones used in the original paper. Nevertheless, we show the goal completion percentage results obtained with our reproduced evaluation compared to DISCERN results from the original paper. LEXA results were obtained with early stopping, and shows our agent solving many tasks in the benchmark.

Table 2.3: Success for DISCERN goals [414].

Task	LEXA	DISCERN
Cup	**84.0%**	76.5%
Cartpole	**35.9%**	21.3%
Finger	**40.9%**	21.8%
Pendulum	**79.1%**	75.7%
Pointmass	**83.2%**	49.6%
Reacher	**100.0%**	87.1%

Plan2Explore Benchmark: We provide a comparison on the standard reward-based DM control tasks [391] in Tab. 2.4. To compare on this benchmark, we create goal images that correspond to the reward functions. This setup is arguably harder for our agent, but is much more practical. Note our agent never observes the reward function and only observes the goal at test time. Plan2Explore adapts to new tasks but it needs the reward function to be known at test time, while DrQV2 is an oracle agent that observes the reward at training time. Baseline results are

Table 2.4: Zero-shot return,P2E tasks [349].

Task	LEXA	P2E	DrQv2
Zero-Shot	✓	✓*	✗
Walker Stand	**957**	331	968
Hopper Stand	**840**	**841**	957
Cartpole Balance	886	**950**	989
Cartpole Bal. Sparse	**996**	860	983
Pendulum Swing Up	**788**	**792**	837
Cup Catch	**969**	**962**	909
Reacher Hard	**937**	66	970

taken from [349, 441]. LEXA results were obtained with early stopping. LEXA outperforms Plan2Explore on most tasks and even performs comparably to state of the art oracle agents (DrQ, DrQv2, Dreamer) that use task rewards for training.

2.3.4 Analysis

Prior work: Most work we compared against struggles with exploration, such as SkewFit and DLL methods. DIAYN is augmented with our explorer, but still fails to leverage the exploration data to learn a diverse set of skills. GCSL struggles to fit the exploration data and produces behavior that does not solve the task, perhaps because the exploration data is too diverse. We observed that all baselines make progress on the simple reaching task, but struggle with other tasks. We have experimented with several versions and improvements to the baselines and report the best obtained performance.

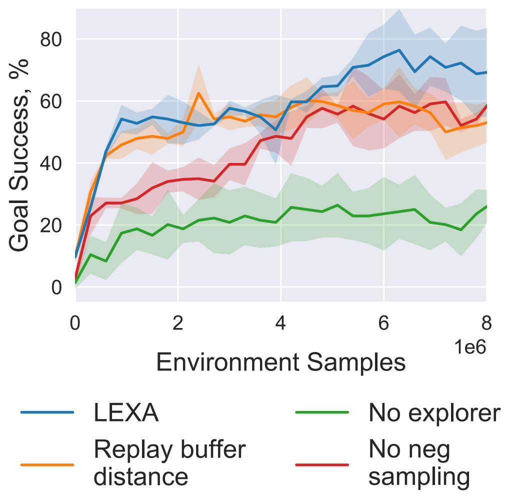


Figure 2.8: Ablations on RoboBins. A separate explorer is crucial for most tasks. Training temporal distance on negative samples speeds up learning, and both negative sampling and training in imagination as opposed to real data are important for the hardest tasks.

Ablation of different components: We ablated components of LEXA on the RoboBins environment in Figure 2.8. Using a separate explorer policy crucial as without it the agent does not discover the more interesting tasks. Without negative sampling the agent learns slower, perhaps because the distance function doesn't produce reasonable outputs when queried on images that are more than horizon length apart. Training the distance function with real data converges to slightly lower success than using imagination data, since real data is sampled in an off-policy manner due to its limited quantity.

Exploration performance: Due to importance of exploration, we further examine the diversity of the data collected during training. We log the instances where the agent coincidentally solves an evaluation task during exploration, for the RoboKitchen and RoboBins environments. In Figure 2.5, we see that our method encounters harder tasks involving multiple objects much more often.

2.4 Related Work

Learning to Achieve Goals: The problem of learning to reach many different goals has been commonly addressed with model-free methods that learn a single goal-conditioned policy [181, 341, 8]. Recent work has combined these approaches with various ways to generate training goals, such as asymmetric self-play [382, 275] or by sampling goals of intermediate difficulty [109, 99]. These can achieve remarkable performance in simulated robotic domains, however, they focus on the settings where the agent can directly perceive the low-dimensional environment state.

A few works have attempted to scale these model-free methods to visual goals by using contrastive [414] or reconstructive [271, 295] representation learning. However, these approaches struggle to perform meaningful exploration as no clear reward signal is available to guide the agent toward solving interesting tasks. Some works [55, 393] avoid this challenge by using a large dataset of interesting behaviors. Other works [295, 457] attempt to explore by generating goals similar to those that have already been seen, but do not try to explore truly novel states.

A particularly relevant set of approaches used model-based methods to achieve goals via planning [107, 97] or learning model-regularized policies [286]. However, these approaches are limited by short planning horizons. In contrast, we learn long-horizon goal-conditioned value functions which allows us to solve more challenging tasks. More generally, most of the above approaches are limited by simplistic exploration, while our method leverages model imagination to search for novel states, which significantly improves exploration and in turn downstream capabilities.

Learning Distance Functions: A crucial challenge for visual goal reaching is the choice of the reward or the cost function for the goal achieving policy. Several approaches use representation learning to create a distance in the feature space [417, 271, 414, 48]. However, this naive distance may not be most reflective of how hard a particular goal is to reach. One line of research has proposed using the mutual information between the current state and the goal as the distance metric [130, 102, 1, 67], however, it remains to be seen whether this approach can scale to more complex tasks.

Other works proposed temporal distances that measure the amount of time it takes to reach the goal. One approach is to learn the distance with approximate dynamic programming using Q-learning methods [181, 110, 103]. Our distance function is most similar to [155], who learn a temporal distance with supervised learning on recent policy experience. In contrast to [155], we always train the distance on-policy in imagination, and we further integrate this achiever policy into our latent explorer achiever framework to discover novel goals for the achiever to practice on.

2.5 Conclusion

We presented Latent Explorer Achiever (LEXA), an agent for unsupervised RL that explores its environment, learns to achieve the discovered goals, and solves image-based tasks in a zero-shot way. By planning for novelty in imagination, LEXA prospectively explores to discover meaningful behaviors in substantially more diverse environments than considered by prior work. Further, LEXA is able to solve challenging downstream tasks specified as images without any supervision such as rewards or demonstrations. By proposing a challenging benchmark and the first agent to achieve meaningful performance on these tasks, we hope to stimulate future research on unsupervised agents, which we believe are fundamentally more scalable than traditional agents that require human design for tasks and rewards.

Many challenges remain for building unsupervised agents. Many tasks in our benchmark are still unsolved and there remains room for progress on the algorithmic side both for the world model and policy optimization. Further, it is important to demonstrate the benefits of unsupervised agents on real-world systems to verify their scalability. Finally, for widespread adoption, it is crucial to design methods that act on goals that are easy to specify, such as via natural language. We believe LEXA will enable future work to tackle these goals effectively.

Chapter 3

ALAN: Autonomously Exploring Robotic Agents in the Real World

3.1 Motivation

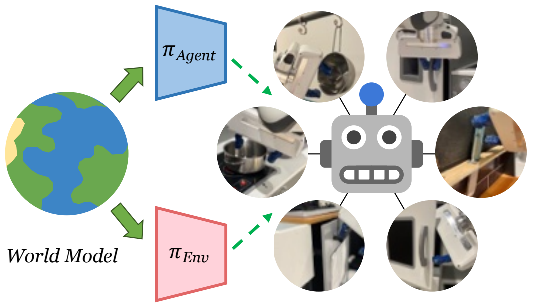


Figure 3.1: We present ALAN, an approach for real world robotic exploration in challenging manipulation environments.

Autonomous robots will need to perform a diverse range of tasks in the real world. Due to the challenges of dealing with uncertainty, deep learning has emerged as a promising approach [221, 294, 182] for robotics. A critical challenge for scaling learning based approaches to more complex settings is the task specification problem. Prior works require heavy reward engineering or human demonstrations, which is cumbersome to obtain for performing large numbers of tasks [283, 317, 176]. This also requires knowledge of the environment, which might be hard to obtain for every domain. Instead, if robots can collect their own data using task-agnostic objectives, they can autonomously explore their environments to learn skills.

In the absence of explicit task definitions, the agent should have an efficient way to use all its collected experience for learning. World models [140, 146] provide a means of learning an effective low dimensional representation of raw image observations. Furthermore, if there are certain states where prediction for the world model is difficult, then it likely needs more data for the corresponding part of the environment. This gives rise to a natural intrinsic objective of maximizing model uncertainty [287, 349, 250] for exploration. While this does lead to the discovery of interesting behavior, there has been difficulty in scaling such approaches to real world settings since collecting samples on real hardware is very time-intensive. We ask if there is a

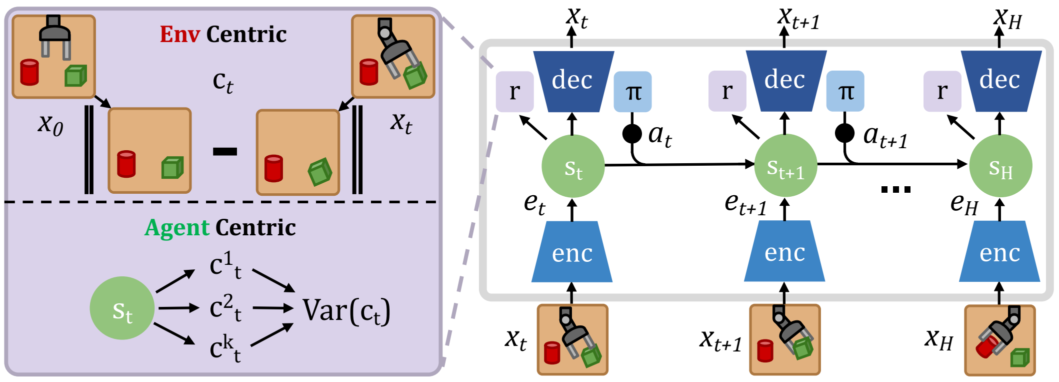


Figure 3.2: We propose Autonomous Learning Agents (ALAN) that can enable robots to collect rich data from their environment efficiently. The agent utilizes environment change, both directly as an environment-centric signal, as well as modelling the change and taking actions that maximize uncertainty in change space, which provides agent-centric signal.

different task-agnostic objective that can enable robots to *more efficiently* explore?

In order to address the above question, we present ALAN, an efficient autonomous real robot explorer. Our key insight is that interesting behavior for robots in the manipulation setting mostly involve *interactions with objects, which cause changes in the visual features of the observations*. Thus, seeking to maximize the change in these visual features can be a useful objective for robots to optimize. Furthermore, if agents learned to model the change in the environment, they can take actions to maximize uncertainty in the *object space* of the environment, as opposed to the full space consisting of both the robot body and the surrounding environment. Seeking to maximize information related to objects in the environment will lead to much more efficient exploration, since the robot will prioritize actions that lead to richer contact interactions. We note that maximizing model uncertainty, (whether in the object space or full image space) is 'agent-centric', since it is dependent on the agent's belief, as opposed to simply maximizing the environment change which is 'environment centric'. The latter is a constant signal agnostic of the agent's internal mental model. We show that leveraging both these objectives can enable a real robot to explore multiple challenging real-world environments, and then perform tasks of interest.

The main contribution of this work is ALAN, an efficient real world exploration algorithm, that seeks to take actions that maximize change in the environment, and maximize uncertainty about its internal model of how changes occur in the environment. This approach encourages the robot to interact with objects, and collect data relevant to learning manipulation skills faster. We show that our approach enables a Franka Emika robot to effectively explore without any supervision signal in two different, challenging play-kitchen environments using less than 150 trajectories. The robot can then perform user-specified tasks via goal images in a zero-shot manner, including picking up a knife, and opening a cabinet, fridge or shelf.

3.2 Related Work

Exploration: In reinforcement learning (RL), exploration has been studied in various contexts ranging from tabular settings to high-dimensional continuous spaces. For simple discrete settings, analysis of exploration has included state visitation counts [379] and probability distributions over visited states [96, 296]. For high-dimensional input spaces such as images, previous works have used neural networks to approximate state counts [28, 277, 390] and for sampling goals [457, 295]. Another approach to describe intrinsic reward for exploration is to use either the error [285] or uncertainty [240, 276] in prediction about how the environment and agent would interact. Pathak et al. [287] proposes a differentiable intrinsic reward which measures disagreement using the variance of the prediction of an ensemble of models. Sekar et al. [349] leverages a similar disagreement-based intrinsic reward, but explores in the imagination space of a learned world model [145, 146].

Autonomous Learning in the Real World: Training agents in the real world is challenging for a host of reasons, and one of these is the difficulty of providing supervision to the agent. Some prior approaches have designed task specific rewards [220, 221]. However, it is infeasible to define all of the tasks that are possible for the robot to perform, and further there is no guarantee that the designed rewards will allow for the task to be solved efficiently and robustly. There are a number of approaches that provide self-supervision for agents based on mutual information objectives [102, 358, 414], which enables the learning of skill-spaces. However, many of these learned skills are not semantically different and have been difficult to apply to real-world manipulation. Other approaches involve selecting goals from experience. This can directly come from previously seen states [8], from a generative model [271, 295, 457], or from the imagination space of a world-model [250]. While these approaches have shown better results for real-world manipulation, they are still limited in scope, since they require lots of samples for learning. A key reason is that it is difficult for the robot to know *what* to focus on while exploring. Efforts have been made to initialize such approaches from priors of human behavior, such as from internet data [357, 19, 56], however, such methods are not able to learn in an autonomous fashion. Our approach provides an effective new metric that enables efficient self-supervision, and also leverages visual priors to focus on parts of the scene that are more interesting for exploration and discovery of useful skills.

3.3 Background

Model-Based RL and Planning: A Markov Decision Process (MDP) is defined by a set of states $\mathcal{S}$, actions $\mathcal{A}$, transition probabilities between states conditioned on actions, $\mathcal{T}(s_{t+1}|s_t, a_t)$, a initial state distribution $\mathcal{S}_0$, a reward function $\mathcal{R}(s_t, a_t)$. The goal of a model based RL algorithm is to learn a function $f_\theta(s_{t+1}|s_t, a_t)$ which best approximates the the true transition dynamics $\mathcal{T}$ of the MDP. While planning, the

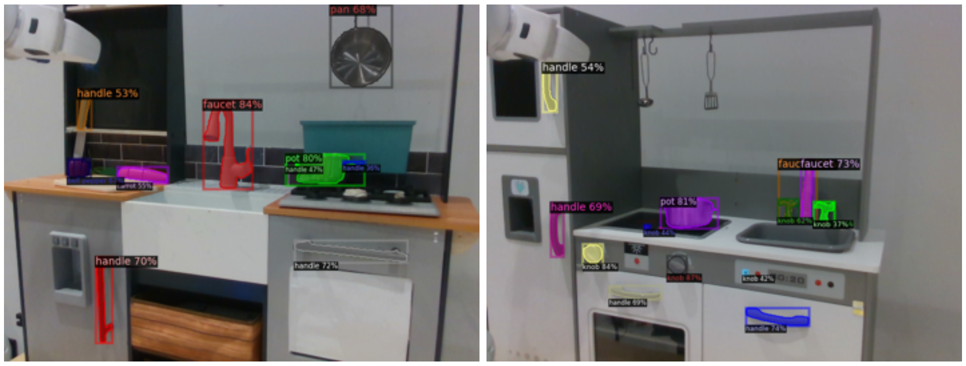


Figure 3.3: Visualizations of the object detections, using [462]. The masks selected to study exploration are the knife, pan and rightcabinet handle from kitchen1 (left), and the topshelf, fridge handles and pot from kitchen2 (right).

Cross-Entropy Method (CEM) can be used to find the best set of actions $a_{1:T}$, which produce the highest reward under the trained dynamics model f_θ.

Intrinsic Motivation: When learning a dynamics model of the world, $f_\theta(s_{t+1}|s_t, a_t)$, it is possible to use the quality of the model as an intrinsic reward. For instance, Pathak et al. [285] use model prediction error as reward

$$r_t = ||f_\theta(s_{t+1}|s_t, a_t) - s_{t+1}||$$

However, this formulation is dependent on environment dynamics, and thus needs a policy-gradient approach to optimize it, since future states need to be observed before this metric can be computed. Instead, [287] proposes to minimize the *disagreement* between an ensemble of dynamics model $f_{\theta^{(k)}}$ for $k = 1, ..., M$, which is a fully differentiable objective in terms of the current state and action, which we utilize in our work. The disagreement reward can be described as:

$$\mathbb{E}_{s_t, a_t, s_{t+1} \sim \rho(s)}[\text{Var}_k(f_{\theta^{(k)}})]$$

3.4 Autonomous Real World Robot Learning

Intelligent agents should be able to perform diverse tasks in complex, real world environments. There are three major challenges to this: (1) There is a large space of possible interactions, especially in continuous control. (2) It is difficult to obtain any reward signal without human supervision. (3) There is a large cost for collecting data with real hardware.

To this end, we propose ALAN, an autonomous robot learning algorithm that is able to efficiently explore in the real world, and learn useful manipulation skills for various objects. ALAN defines a novel intrinsic exploration objective for the agent to direct its behavior. This novel objective has an environment-centric component and an agent-centric component. Moreover, we use offline visual data to reduce the search space for the robot, by identifying the locations of potential interesting and complex interactions for the robot.

3.4.1 World Model

The robot observations consist of a stream of high-dimensional raw RGB images. These can be effectively processed using world models [140, 344, 343], which learn compact low-dimensional latent spaces that contain temporal information and enable efficient forward prediction. We use the Recurrent state-space model (RSSM), from [145, 146, 147], which learns latent features with deterministic and stochastic components to model long-range dependencies and uncertainty in the environment respectively. Specifically, the world model has the following networks:

Image Encoder	$h_t = \text{enc}_\theta(x_t)$
Dynamics Prior	$p_\theta(s_{t+1} \mid s_t, a_t)$
Image Decoder	$f_\theta(x_t \mid s_t)$
Dynamics Posterior	$q_\theta(s_{t+1} \mid s_t, a_t, h_{t+1})$
Embed Decoder	$g_\theta(e_t \mid s_t)$

The latent features are trained to reconstruct image observations, while also preserving dynamics information using variational inference and the ELBO loss [323, 196]. In addition to providing useful representations for control, world models also provides a means for agents to drive their own behavior in the absence of supervision. This involves taking actions that maximize the uncertainty of model predictions [287, 349, 250], leading to information gain for the agent. Since this is dependent on the agent's internal belief, we call this kind of exploration 'agent-centric'. In the next section we first consider a different source of signal which is environment-centric, and then discuss how it can be used to augment agent-centric exploration as well.

3.4.2 Environment Change

Seeing as how interesting manipulation behavior often involves changes in object states, and how this corresponds to change in visual features, we seek to autonomously estimate environment change from observed data. To capture environment interaction, the change metric should ignore differences in the robot's position, and only highlight movement of objects in the scene [19]. How then can we extract these ground truth change images from incoming image observations?

Our source of signal is assuming knowledge of the visual appearance of the robot, using which we train a segmentation model $m_\phi(.)$ to mask out the robot from the scene. Training this model is a one time cost, since the robot appearance is invariant across multiple tasks in the environment and even across different domains. We can use this model to measure the environment change f_c between an image pair x_i, x_j:

$$f_c(x_i, x_j) = f(||m_\phi(x_i) - m_\phi(x_j)||_2, \\ ||\Psi(m_\phi(x_i)) - \Psi(m_\phi(x_j))||_2) \tag{3.1}$$

Here the heuristic function f takes into account pixel distance, blurring to remove shadows and reflective surface artifacts, and Ψ denotes visual features from a pretrained segmentation network [158], and returns a binary image indicating the pixels where change has been detected. We further apply a threshold for the change image, in order to minimize false detections. We don't require this change function to be fully accurate, and have found that our approach is robust to some error in the change image. For an image x_t from a trajectory $\mathcal{T}$, the corresponding change c_t can be defined as $f_c(x_t, x_{t-1})$ or $f_c(x_t, x_0)$, where x_0 is the first image in $\mathcal{T}$. We found the latter produced better exploration, likely because the change between consecutive image frames is very small and is difficult to reliably detect.

Algorithm 2 ALAN : Exploration

```
Require: Robot segmentation model $m_\phi$
Require: Off-policy RL algorithm $\mathcal{A}$
Require: Visual Priors (3.4.3)
 1: Initialize: World Model $\mathcal{W}$, Biasing policy $\pi$,
 2: Initialize: Dataset $R_D$
 3: while Sampling do
 4:     Run $\pi$ with $\mathcal{W}$ in imagination to get $\{\hat{a}_t\}_H$
 5:     Run CEM with $\mathcal{W}$ using Eqs.(3.2, 3.3),and
        initial proposal $\{\hat{a}_t\}_H$, to get trajectory $\mathcal{T}$
 6:     Label $\mathcal{T}$ with $c_t = f_c(x_t, x_0)$ (Eq. L.1)
 7:     Add $\mathcal{T}$ to $R_D$
 8: end while
 9: while Training do
10:     $\mathcal{S_D}$ = Top $N_A$ trajs in $\mathcal{R_D}$, based on $\sum c_t$
11:     Update $\pi$ using $\mathcal{A}$ on $\mathcal{S_D}$
12:     Update $\mathcal{W}$ using $R_D$
13: end while
```

Env-centric exploration: Using the norm of the change image as a metric, we can use off-policy RL [289, 272, 203] approaches to train a policy for control. The approach we use is to incorporate the metric into a world model by training the features s_t to also predict the change in the environment between observation o_t and the initial observation of the trajectory o_0, by adding an additional change predictor module $r_\theta(c_t|s_t)$. This is optimized by maximizing $\mathbb{E}[\log p(c_t|s_t)]$, similar to the image

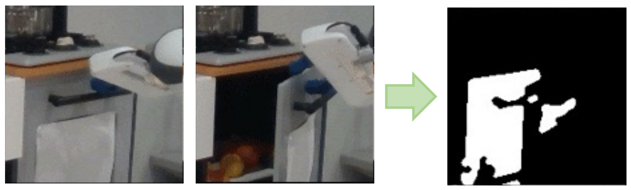


Figure 3.4: Change image extracted from a pair of images, as described in Eq. L.1. This is a binary image that detects pixels where change has occurred.

decoder, where c_t is the change image. Specifically, we optimize:

$$\arg\max_{a_1..a_T} \mathbb{E}_{s\sim\rho(s)}[\sum(r_\theta(c_{t+1}|s_{t+1})\big|s_t, a_t)] \tag{3.2}$$

Change-space agent-centric exploration: Since the agent now models the environment change in its internal belief, it can leverage errors in this model to direct exploration. Just as previous exploration approaches maximize uncertainty of next state using the model [287, 349] the agent can maximize uncertainty over the *change* prediction. Thus, the agent will collect data that leads to information gain specifically about how the objects in the environment move, avoiding being stuck gathering information pertaining to the robot's own body. Thus the agent will collect data that includes more information about object interactions. Specifically, we implement this by training an ensemble of models for $p(c_{t+1}|c_t, a_t)$, where c_t and a_t are the predicted change and action at time t respectively. To maximize uncertainty in change space, we optimize for actions that maximize the variance of the ensemble prediction (here s_t is a latent sampled from the world model) :

$$\arg\max_{a_1..a_T} \mathbb{E}_{s\sim\rho(s)}[\mathrm{Var}_k(p_{\psi^{(k)}}(r_\theta(c_{t+1}|s_{t+1}))\big|r_\theta(c_t|s_t), a_t)] \tag{3.3}$$

Control: Now that the features of the world model are trained to predict environment change, we can explore by planning through the model adding the objectives from 3.3 and 3.2. We use the Cross entropy method [332] for planning, where we sample action proposals from an initial distribution, pick the top trajectories based on reward and refit the sampling distribution. Further, we train Advantage Weighted Regression (AWR) on the collected offline trajectories to maximize the environment change in the feature space of the world model. When sampling, given an observation, we first run the learned AWR policy through the model in imagination to get a sequence of actions. We use this as the mean of the initial normal sampling distribution for CEM, to bias the optimization procedure towards trajectories that are likely to have high environment change. We summarize the full exploration method in Alg. 2, including both sampling and training which are run asynchronously.

3.4.3 Leveraging Visual Priors

While environment change and ensemble disagreement can provide useful signal for driving behavior, the large work spaces in the real world pose a major challenge for robots. Exploration methods often spend a lot of time in free space, and collect a large number of samples without interacting with any objects. This is undesirable since this data contributes little to learning manipulation skills. Our approach to avoiding this is to leverage visual priors from offline data, helping understand *what* to explore. One instance of this is to leverage object-detectors to initialize the robot near regions of interest. Recent models [462] are quite robust and can identify objects even in cluttered scenes. Using RGBD cameras and homography calibration for the robot with the cameras, we can then initialize the robot end effector close to the center of the object point-cloud, thus ensuring that data-collection is more likely to see object interactions. This approach does not preclude training on undetected objects, since the robot can always randomly sample points in the full workspace to initialize at later, and will likely be more proficient after it has learned skills efficiently on all the detected objects. For a image that has k detected masks $M_1, ..., M_k$, the robot can arbitrarily pick any mask for initialization every episode. However, in order to study exploration for independent objects separately, we enforce that the robot needs to reset to the same mask each time, and since this choice can be arbitrary, we also specify which mask should be selected, so that different methods can be evaluated on the same objects. We use the same visual prior for the baselines and ablations to make the exploration space feasible.

3.4.4 Achieving goals

Given the contact-rich data collected by the exploration controllers, how can we use this data to perform useful tasks? It is possible for the agent to sample goals from previously seen exploration data. Since the agent sees interesting data, any possible state can be a goal. Concretely, given some human sampled goal images, x_g, we leverage recent advances in goal-conditioned imitation learning, especially methods that leverage Nearest Neighbor-based techniques in a self-supervised representation space [279]. Our policy, π_{knn} scans through image features [273] in the exploratory data, and selects the top trajectory matches:

$$\tau^\star = \text{argmin}_i \min_{x_j \in \tau_i} ||\phi(x_g) - \phi(x_j)||_2 \tag{3.4}$$

Since our method sees interesting trajectories, it is more likely to see semantically useful goals, and when a human provided goal x_{gh} is given, more likely to reach it.

3.5 Experimental Setup

In our experiments, we ask the following questions : 1) Does our system enable autonomous exploration and discovery of interesting states in complex real world environments? 2) How does the quality of this data compare to that of current SOTA approaches? 3) Is it possible to use this data to reach human specified goals to perform useful tasks?

Real World Setup: We tested our system on a Franka Panda 7-DOF robot, and on two different real-world kitchen play-sets, which have many diverse objects and possible manipulation tasks, comprising a very large search space (both are about 100cm X 100cm X 100cm). Specifically, we investigate 6 object regions across two kitchens detected by our visual prior approach [462], as shown in Figure 3.3. Namely, these are the knife, cabinet and the hanging pan from the first kitchen, and the top shelf, fridge and pot from the second kitchen (Figure 3.5).

Training Procedure: For each of the regions, we first collect a random dataset of 25 trajectories. All collected trajectories are 20 timesteps long. The world models in all methods use an RSSM [145], and the image encoders and decoders use the NVAE architecture [400]. To extract the environment centric metric, we train a Mask RCNN model [158] on 200 images from both play kitchens.
Baselines: We compare against LEXA[250], a state-of-the art self-supervised exploration approach for continuous control in manipulation settings. LEXA outperforms various other self-supervised approaches, [295, 102, 155] on a complex simulated kitchen environment both in terms of the exploratory data seen, and the success rate of reaching discovered goal images. We provide this baseline with the same world model architecture as ALAN. Next, we ablate the need of our agent-centric module, which explores in the change space. This is to test our hypothesis that the robot should continually collect data where the model predictions regarding environment change are inaccurate. We test if this ability is crucial, by running the environment-centric exploration model, which only uses the intrinsic reward described in Equation L.1. We run two versions of this, EC which uses the model for planning, and AWR which just uses the trained AWR policy.

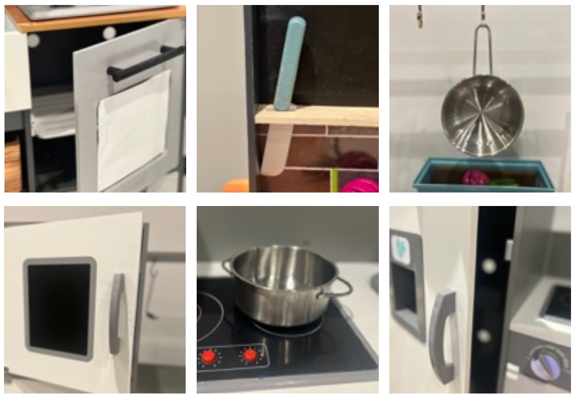


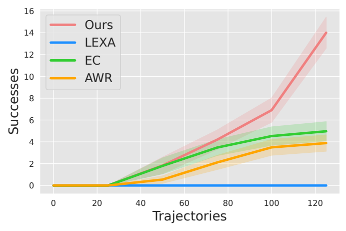
Figure 3.5: We explore on 6 settings across two play kitchens. Top: cabinet, knife, pan (kitchen1). Bottom: top shelf, pot, fridge (kitchen2).

3.6 Results

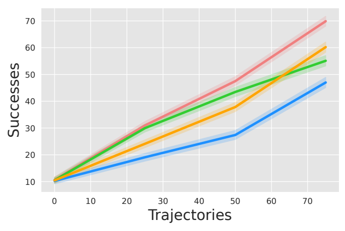
3.6.1 Exploration

We need a metric to evaluate the quality of the exploration data. While the change image norm is a good proxy for measuring object interaction, it does not consider if the different states are semantically interesting. Thus we define a metric that measures the *number* of successful interactions, which are are determined by a human operator, as follows :

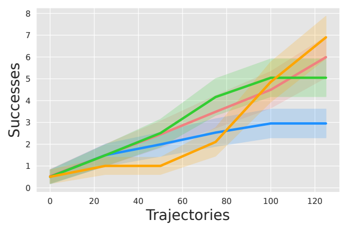
- Cabinet, fridge, shelf doors - has been opened or closed
- Knife - lifted up
- Pan - unhooked, fully removed from hanger
- Pot - pushed/lifted/knocked over



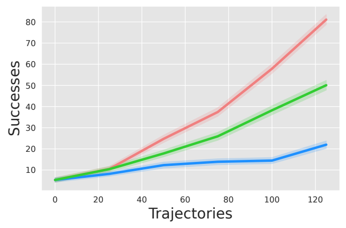
(a) Knife



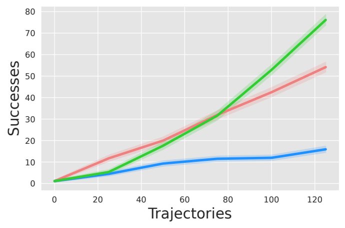
(b) Cabinet



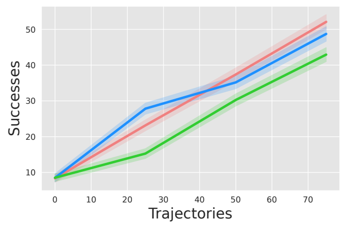
(c) Pan



(d) Fridge



(e) Top Shelf



(f) Stovepot

Figure 3.6: Coincidental success for exploration on our six tasks, where the robot reaches a semantically meaningful state while collecting data during exploration. We can see that ALAN performs consistently well across tasks, and that just maximizing the change metric `AWR`, `EC` also yields much better data than previous state of the art approach `LEXA`.

Using this success criteria, we present evaluation of the exploratory data collected, in Figure 3.6. For each task we run about 100-150 trajectories, and plot the cumulative number of successful exploration trajectories against the total number of trajectories seen during the exploration phase.

We can see that ALAN (red) outperforms or matches all other approaches in five out of six tasks, and also sees large number of successes for the top shelf. Further, we see that just maximizing the environment-change metric using `EC` or `AWR` leads to much better performance than `LEXA`, the previous state-of-the-art self-supervised exploration approach. We find that because the robot arm takes up a large portion of the observation, `LEXA` tries to collect data to resolve modelling inaccuracies of the arm. This is especially the case for tasks where random interactions are less likely to produce significant changes in the object, such as the particularly challenging knife task where `LEXA` never sees the picking up behavior. On this task, having the agent-centric module which maximizes uncertainty in change space significantly improves performance over `EC` and `AWR`. For tasks like the top shelf which require less precise control, simply maximizing environment change is sufficient to collect high-quality data. However, even with slightly more involved control, such as the fridge task which requires the same object motion but has the robot in a more constrained position, addressing modelling inaccuracies in the change prediction is more critical.

	Cabinet	**Knife**	**Fridge**	**Top Shelf**
LEXA [250]	0.20	0.00	0.00	0.00
EC	0.70	0.00	0.50	**0.90**
AWR [289]	0.50	0.00	-	-
ALAN (ours)	**1.00**	**0.60**	**0.70**	0.80

Table 3.1: Success rate for goal reaching. ALAN is the only approach to get success on the challenging knife pick-up task, and just maximizing change (EC) is also stronger than LEXA.

3.6.2 Achieving Goals

Given the exploration data collected, can it be used to perform useful human specified tasks? For this, we use the nearest-neighbor (kNN) approach outlined in section 3.4.4, paired with model-based refinement to reach different human-specified goals. Specifically, once the kNN approach finds a trajectory, we use the action sequence as the mean of the initial sampling distribution of the CEM optimizer. The goals consist of a fully open fridge, cabinet or shelf, and a picked-up knife, as shown in Figure 3.7. Since AWR has almost identical results for exploration and goal-reaching to EC on the first kitchen, and since they both optimize the same objective, we did not run it on the second kitchen. For each task, we run kNN on the exploratory data, in a visual feature space [273] and select the best trajectory to execute conditioned on the start and goal images. We execute the top two trajectories five times each, collecting 10 different trials and present average success rates in Table 3.1. Without the agent-centric module, there is no success on the difficult knife task, and performance across the remaining tasks is worse in terms of robustness. Moreover these results show the effectiveness of the environment change metric, since LEXA shows no success for three of the four tasks.

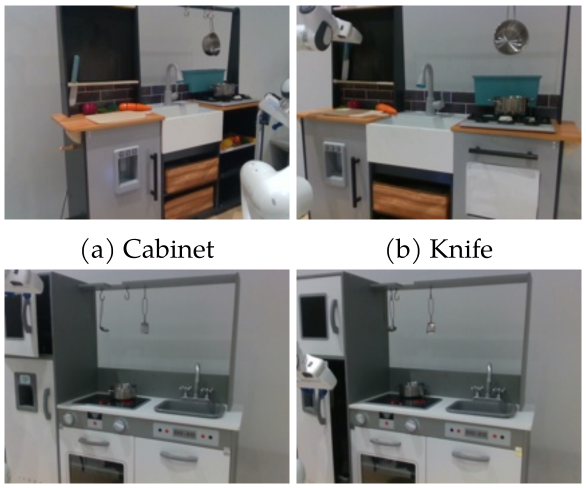

(a) Cabinet (b) Knife (c) Top Shelf (d) Fridge

Figure 3.7: Goals used for zero-shot evaluation, after the completion of the exploration phase.

3.7 Discussion and Limitations

We present ALAN, an autonomously exploring agent that can efficiently explore in challenging real world environments. Our approach computes change in the environment, and utilizes it both directly as an environment-centric signal, as well as modelling the change and taking actions that maximize uncertainty in change space, which provides agent-centric signal. This reward in the absence of true task rewards helps our agent autonomously discover manipulation skills and perform useful tasks without any supervision. In the future, we hope to investigate distilling exploration data into a general goal-achieving policy, and studying continual learning across different tasks using a joint world model.

Part II

Generalist Explorers using Prior Video

Chapter 4

Affordances from Human Videos as a Versatile Representation for Robotics

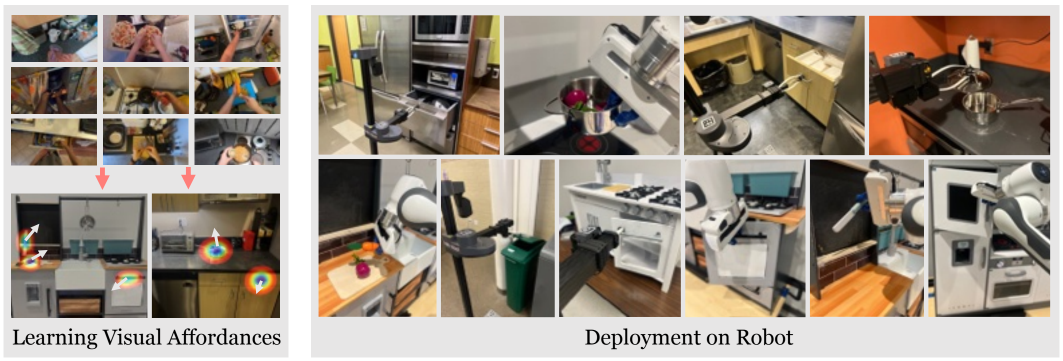


Figure 4.1: We leverage human videos to learn visual affordances that can be deployed on multiple real robot, in the wild, spanning several tasks and learning paradigms.

The meaning or value of a thing consists of what it affords... what we perceive when we look at objects are their affordances, not their qualities

J.J. Gibson (1979)

4.1 Motivation

Imagine standing in a brand-new kitchen. Before taking even a single action, we already have a good understanding of how most objects should be manipulated. This understanding goes beyond semantics as we have a belief of where to hold objects and which direction to move them in, allowing us to interact with it. For instance, the oven is opened by pulling the handle downwards, the tap should be

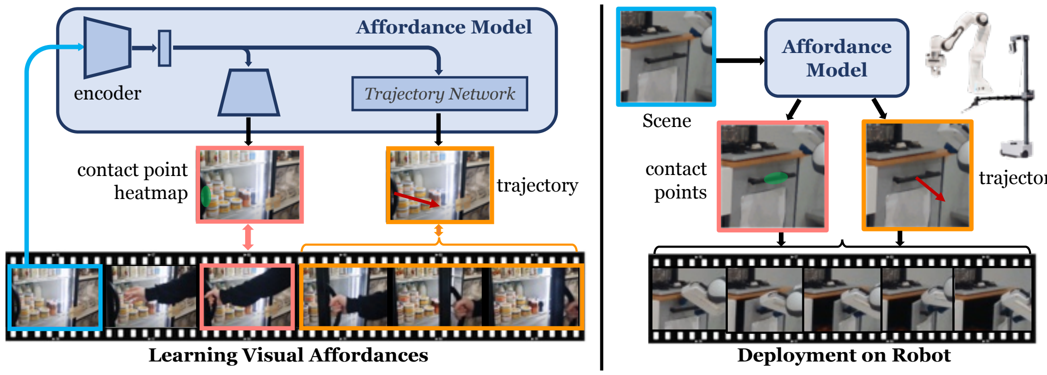


Figure 4.2: **VRB Overview:** First, we learn an actionable representation of visual affordances from human videos: the model predicts contact points and trajectory waypoints with supervision from future frames. For robot deployment, we query the affordance model and convert its outputs to 3D actions to execute.

turned sideways, drawers are to be pulled outwards, and light switches are turned on with a flick. While things don't always work as imagined and some exploration might be needed, but humans heavily rely on such visual *affordances* of objects to efficiently perform day-to-day tasks across environments [124, 125]. Extracting such actionable knowledge from videos has long inspired the vision community.
More recently, with improving performance on static datasets, the field is increasingly adopting a broader 'active' definition of vision through research in egocentric visual understanding and visual affordances from videos of human interaction. With deep learning, methods can now predict heatmaps of where a human would interact [267, 128] or segmentation of the object being interacted with [356]. Despite being motivated by the goal of enabling downstream robotic tasks, prior methods for affordance learning are tested primarily on human video datasets with no physical robot or in-the-wild experiments. Without integration with a robotic system, even the most basic question of how the affordance should be defined or represented remains unanswered, let alone evaluating its performance.
On the contrary, most robot learning approaches, whether imitation or reinforcement learning, approach a new task or a new environment *tabula rasa*. At best, the visual representation might be pretrained on some dataset [355, 273, 241, 429, 309, 436]. However, visual representations are only a small part of the larger problem. In robotics, especially in continuous control, the state space complexity grows exponentially with actions. Thus, even with perfect perception, knowing what to do is difficult. Given an image, current computer vision approaches can label most of the objects, and even tell us approximately where they are but this is not sufficient for the robot to perform the task. It also needs to know *where* and how to manipulate the object, and figuring this out from scratch in every new environment is virtually impossible for all but the simplest of tasks. How do we alleviate this clear gap between visual learning and robotics?

In this chapter, we propose to rethink visual affordances as a means to bridge vision and robotics. We argue that rich video datasets of humans interacting can offer a lot more actionable information beyond just replacing ImageNet as a pretrained visual encoder for robot learning. Particularly, human interactions are a rich source of how a wide range of objects can be held and what are useful ways to manipulate their state. However, several challenges hinder the smooth integration of vision and robotics. We group them into three parts. *First*, what is an actionable way to represent affordances? *Second*, how to learn this representation in a data-driven and scalable manner? *Third*, how to adapt visual affordances for deployment across robot learning paradigms? To answer the first question, we find that contact points and interaction directions are excellent robot-centric representations of visual affordances, as well as modeling the inherent multi-modality of possible interactions. We make effective use of egocentric datasets in order to tackle the second question. In particular, we reformulate the data to focus on frames without humans for predicting contact points and the interaction directions. To extract free supervision for this prediction, we utilize off-the-shelf tools for estimating egomotion, human pose, and hand-object interaction. Finally, we show how to seamlessly integrate these affordance priors with different kinds of robot learning paradigms. We thus call our approach **V**ision-**R**obotics **B**ridge (VRB) due to its core goal of bridging vision and robotics.
We evaluate both the quality of our affordances and their usefulness for 4 different robotic paradigms – imitation and offline learning, exploration, visual goal-reaching, and using the affordance model as a parameterization for action spaces. These are studied via extensive and rigorous real-world experiments on physical robots which span across 10 real-world tasks, 4 environments, and 2 robot hardware platforms. Many of these tasks are performed *in-the-wild* outside of lab environments (see Figure 4.1). We find that VRB outperforms other state-of-the-art human hand-object affordance models, and enables high-performance robot learning in the wild without requiring any simulation. Finally, we also observe that our affordance model learns a good visual representation for robotics as a byproduct. We highlight that all the evaluations are **performed in the real world spanning several hundred hours of robot running time** which is a very large-scale evaluation in robotics.

4.2 Related Work

Affordance and Interaction Learning from Videos: Given a scene, one can predict interactions using geometry-based rules for objects via 3D scene understanding [156, 459, 264], estimating 3D physical attributes [100, 25, 132, 465] or through segmentation models trained on semantic interactions [330, 339], and thus require specialized datasets. More general interaction information can be learned from large human datasets [223, 76, 77, 238, 80, 129], to predict object information [466, 119] (RGB & 3D) [29], graphs [91] or environment information [268, 111] such as heatmaps [128, 267]. Approaches also track human poses, especially hands

[356, 233, 329, 77, 236, 49, 443]. Similarly, in action anticipation and human motion forecasting, high-level semantic or low level actions are predicted using visual history [201, 324, 121, 171, 166, 405, 105, 76, 81, 172, 40, 117, 212, 404, 129, 118, 256, 249, 126]. Since our observations only have robot arms and no human hands, we adopt a robot-first formulation, only modeling the contact point and post-contact phase of interaction for our affordances.

Visual Robot Learning: Learning control from visual inputs directly is an important challenge. Previous works have leveraged spatial structures of convolutional networks to directly output locations for grasping and pushing from just an image of the scene [294, 452, 453], which can limit the type of tasks possible. It is also possible to directly learn control end-to-end [221, 182] which while general, is quite sample inefficient in the real world. It has been common to introduce some form of prior derived from human knowledge, which could take the form of corrective interactions [139, 239, 84], structured policy spaces [274, 73, 319, 299, 7, 174, 18, 18, 357, 438], offline robotics data [98, 208, 207, 245, 311], using pretrained visual representations [355, 281, 273, 429, 436] or human demonstrations [351, 56, 19, 370, 360, 357].

Learning Manipulation from Humans: Extensive work has been done on Learning from Demonstrations (LfD) where human supervision is usually provided through teleoperation (of a joystick or VR interface) [376, 456, 259] or kinesthetic teaching, where a user physically moves the robot arm [299, 46, 69, 101, 243].With both these approaches, collecting demonstrations is tedious and slow. Recently, works have shown alternate ways to provide human demonstrations, via hand pose estimation and retargeting [369, 12, 442, 361, 302] in robot hands, but are mostly restricted to tabletop setups. First and third person human demonstrations have been used to train policies directly, transferred either via a handheld gripper [375, 446, 279] or using online adaptation [19]. In contrast to directly mimicking a demonstration, we learn robot-centric *affordances* from passive human videos that provide a great initialization for downstream robot tasks, unlike previous work which require in-domain demonstrations.

4.3 Affordances from Human Videos (VRB)

Our goal is to learn affordance priors from large-scale egocentric videos of human interaction, and then use them to expedite robot learning in the wild. This requires addressing the three questions discussed in Sec. 4.1 about how to best represent affordances, how to extract them and how to use them across robot learning paradigms.

4.3.1 Actionable Representation for Affordances

Affordances are only meaningful if there is an actor to execute them. For example, a chair has a sitting affordance only if it is possible for some person to sit on it. This

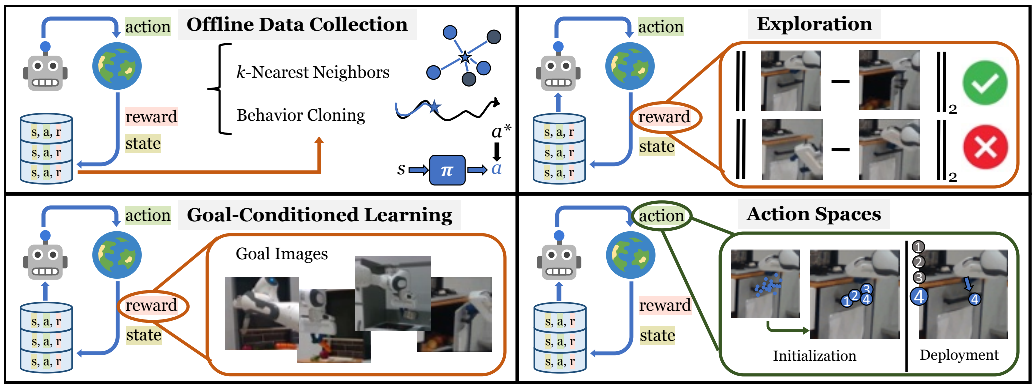


Figure 4.3: **Robot Learning Paradigms:** (a) Offline Data Collection – Used to investigate the quality of the collected data. (b) Exploration – The robot needs to use intrinsic rewards to improve (c) Goal-Conditioned Learning – A desired task is specified via a goal image, used to provide reward. (d) Action Spaces – Reduced action spaces are easier to search and allow for discrete control.

property makes it clear that the most natural way to extract human affordances is by watching how people interact with the world. However, what is the right object-centric representation for affordances: is it a heatmap of where the human makes contact? Is it the pre and postcondition of the object? Is it a description of the human interaction? All of these are correct answers and have been studied in prior works [267, 236, 156]. However, the affordance parameterization should be amenable to deployment on robots.

If we want the robot to *a priori* understand how to manipulate a pan (Fig. 4.1, 4.4) without any interaction, then a seemingly simple solution is to exactly model human movement from videos [236], but this leads to a human-centric model and will not generalize well because human morphology is starkly different from that of robots. Instead, we take a first-principles approach driven by the needs of robot learning. Knowledge of a robot body is often known, hence reaching a point in the 3D space is feasible using motion planning [213, 214, 185]. The difficulty is in figuring out where to interact (e.g. the handle of the lid) and then how to move after the contact is made (e.g., move the lid upwards).

Inspired by this, we adopt contact points and interaction directions as a simple actionable representation of visual affordance that can be easily transferred to robots. We use the notation c for a contact point and τ for interaction direction, both in the pixel space. Specifically, $\tau = f(I_t, h_t)$, where I_t is the image at timestep t, h_t is the human hand location in pixel space, and f is a learned model. We find that our affordance representation outperforms prior formulations across robots. Notably, the c and τ abstraction makes the affordance prior agnostic to the morphological differences across robots.

4.3.2 Learning Affordances from Egocentric Videos

The next question is how to extract c and τ from human videos in a scalable data-driven manner while dealing with the presence of human body or hand in the visual input. VRB tackles this through a robot-first approach.

4.3.2.1 Extracting Affordances from Human Videos

Consider a video V, say of a person opening a door, consisting of T frames *i.e.* $V = \{I_1, ..., I_T\}$. We have a twofold objective — find *where* and *when* the contact happened, and estimate how the hand moved after contact was made. This is used to supervise the predictive model $f_\theta(I_t)$ that outputs contact points and interaction directions. To do so, we utilize a widely-adopted hand-object detection model trained on human video data [356]. For each image I_t, this produces 2D bounding boxes of the hand h_t, and a discrete contact variable o_t. Using this information, we filter for frames where o_t indicates a contact in each video, and find the first timestep where contact occurs, t_{contact}.
The pixel-space positions of the hand $\{h_t\}_{t_{\text{contact}}}^{t'}$ constitute the interaction direction (τ). To extract contact points c, we use the corresponding hand bounding box, and apply skin color segmentation to find all points at the periphery of the hand segment that intersect with the bounding box of the object in contact. This gives us a set of N contact points $\{c^i\}^N$, where N can differ depending on the image, object, scene and type of interaction. How should the contact points be aggregated to train our affordance model (f_θ)? Some options include predicting the mean of $\{c^i\}^N$, or randomly sampling c^i. However, we seek to encourage multi-modality in the predictions, since a scene likely contains multiple possible interactions. To enable this, we fit a Gaussian mixture model (GMM) to the points. Let us define a distribution over contact points to be $p(c)$. We fit the GMM parameters (μ_k, Σ_k) and weights α_k.

$$p(c) = \underset{\mu_1,...,\mu_K,\Sigma_1,...,\Sigma_K}{\arg\max} \sum_{i=1}^{N} \sum_{k=1}^{K} \alpha_k \mathcal{N}(c^i | \mu_k, \Sigma_k) \tag{4.1}$$

We use these parameters of the above defined GMM with K clusters as targets for f_θ. To summarize, 1) we find the first timestep where contact occurs in the human video, t_{contact} 2) For c, we fit a GMM to the contact points around the hand at frame $I_{t_{\text{contact}}}$, parameterized by μ_k, Σ_k and 3) we find the post-contact trajectory of the 2D hand bounding box $\{h_t\}_{t_{\text{contact}}}^{t'}$ for τ.

Accounting for Camera Motion over Time: Consider a person opening a door. Not only do the person's hands move but their body and hence their head also move closer to the handle and then away from it. Therefore, we need to compensate for this egomotion of the human head/camera from time t_{contact} to t'. We address this by using the homography matrix at timestep t, $\mathcal{H}_t$ to project the points back into the coordinates of the starting frame. We obtain the homography matrix by matching features between consecutive frames. We then use this to produce the transformed trajectory $\tau = \mathcal{H}_t \circ \{h_t\}_{t_{\text{contact}}}^{t'}$.

Addressing Human-Robot Visual Domain Shift: The training videos contain human body or hand in the frame but the human will not be present in downstream robotics task, generating domain shift. We deal with this issue with a simple yet elegant trick: we extract affordances in the frames with humans but then map those affordances back to the first frame when human was yet to enter the scene. For videos in which a human is always in frame, we either crop out the human in the initial frame if there is no interaction yet or discard the frame if the human is always in contact. We compute the contact points and interaction directions with respect to this human-less frame via the same homography procedure described above. This human-less frame is then used to condition our affordance model.

4.3.2.2 Training Affordance Model

Conditioned on the input image, the affordance model is trained to predict the extracted labels for contact points and interaction directions. However, naive joint prediction does not work well as the learning problem is inherently multi-modal. For instance, one would pick up a cup differently from a table depending on whether the goal is to pour it into the sink or take a sip from it. We handle this by predicting multiple heatmaps for interaction points using the same model, building a spatial probability distribution.

For ease of notation, we use $(\cdot)_\theta$ as a catch-all for all parameterized modules and use f_θ to denote our complete network. Fig. 4.2 shows an overview of our model. Input image I_t is encoded using a ResNet [157] visual encoder g_θ^{conv} to give a spatial latent representation z_t, i.e., $g_\theta^{\text{conv}}(I_t) = z_t$. We then project this latent z_t into K probability distributions or heatmaps using deconvolutional layers; concretely, $H_t = g_\theta^{\text{deconv}}(z_t)$. Using a spatial softmax, σ_{2D}, we get the estimation of the labels for GMM means, *i.e.*, μ_k. We found that keeping the covariance matrices fixed gave better results. Formally, the loss for contact point estimation is:

$$\mathcal{L}_{\text{contact}} = \left\| \mu_i - \sigma_{\text{2D}}\left(g_\theta^{\text{deconv}}\left(g_\theta^{\text{conv}}\left(I_t\right)\right)\right) \right\|_2 \tag{4.2}$$

To estimate interaction direction, we train a trajectory prediction network, $\mathcal{T}_\theta$, based on the latent representation z_t. We find that it is easier to optimize for *relative* shifts, *i.e.*, the direction of movement instead of absolute locations, assuming that the first point $\hat{w}_0$ is 0, since the contact points are already spatially grounded. Based on the success of Transformers for sequential prediction, we employ self-attention blocks [402] and train to optimize $\mathcal{L}_{\text{traj}} = \|\tau - \mathcal{T}_\theta(z_t)\|_2$. In a given scene, there are many objects a human could interact with, which may or may not be present in the training data. We tackle this uncertainty and avoid spurious correlations by sampling local crops of I_t around the contact points. These serve as the effective input to our network f_θ and enables better generalization.

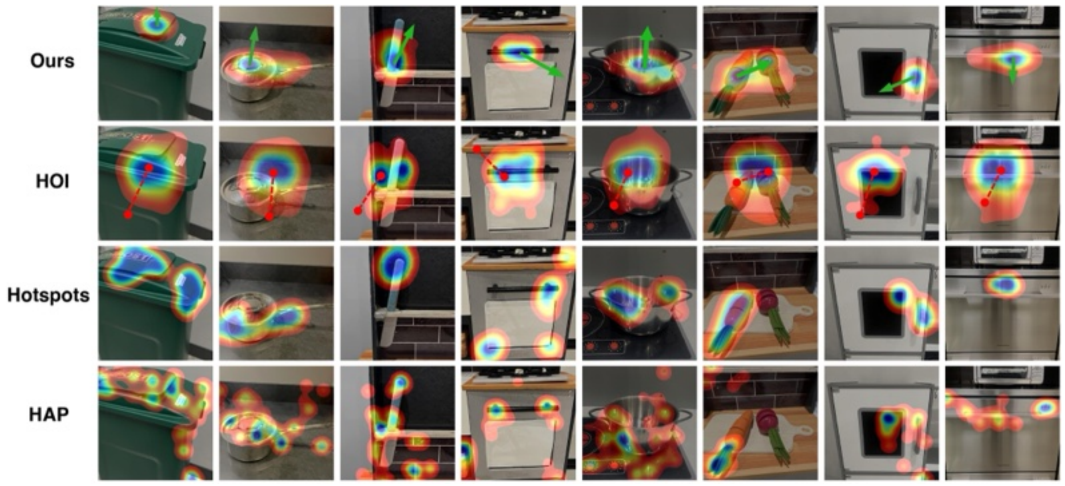


Figure 4.4: Qualitative affordance model outputs for `VRB`, `HOI` [236], `Hotspots` [128] and `HAP` [128], showing the predicted contact point region, and post-grasp trajectory (green arrow for `VRB`, red for `HOI` [236]). We can see that `VRB` produces the most meaningful affordances.

4.3.3 Robot Learning from Visual Affordances

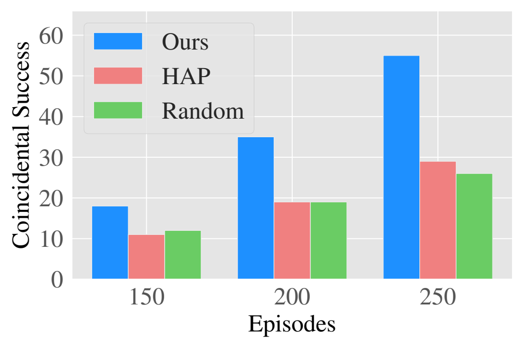
Instead of finding a particular way to use our affordance model for robotics, we show that it can bootstrap existing robot learning methods. In particular, we consider four different robotics paradigms as shown in Fig. 4.3.

A. Imitation Learning from Offline Data Collection: Imitation learning is conventionally performed on data collected by human demonstrations, teleoperation, or scripted policies – all of which are expensive and only allow for small-scale data collection[447, 19, 360, 10, 41, 221]. On the other hand, using the affordance model, $f_\theta(\cdot)$ to guide the robot has a high probability of yielding 'interesting' interactions. Given an image input I_t, the affordance model produces $(c, \tau) = f_\theta(I_t)$, and we store $\{(I_t, (c, \tau))\}$ in a dataset $\mathcal{D}$. After sufficient data has been collected, we can use imitation learning to learn control policies, often to complete a specific task. A common approach for task specification is to use *goal images* that show the desired configuration of objects. Given the goal image, the *k-Nearest Neighbors* (k-NN) approach involves filtering trajectories in $\mathcal{D}$ based on their distance to the goal image in feature space. Further, the top (filtered) trajectories can be used for *behavior cloning* (BC) by training a policy, $\pi(c, \tau | I_t)$. We run both k-NN and behavior cloning on datasets collected by different methods in Sec. 4.4.1. Using the same IL approach for different datasets is also useful for comparing the relative quality of the data. This is because higher relative success for a particular dataset implies that the data is qualitatively better, given that the same IL algorithm achieves worse performance on a different dataset. This indicates that the goal (or similar images) were likely seen during data collection.

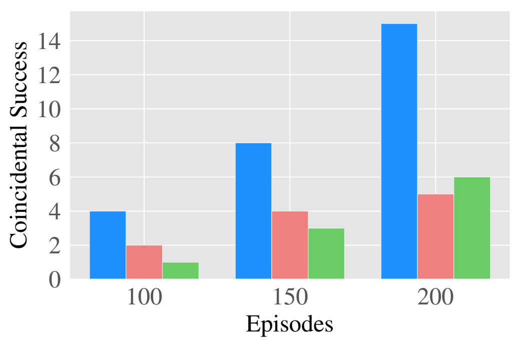
B. Reward-Free Exploration: The goal of exploration is to discover as many diverse skills as possible which can aid the robot in solving downstream tasks. Exploration methods are usually guided by *intrinsic rewards* that are self-generated by the robotic agent, and are not specific to any task [28, 285, 277, 390, 232, 295, 173, 313, 251]. However, starting exploration from scratch is too inefficient in the real world, as the robot can spend an extremely large amount of time trying to explore and still not learn meaningful skills to solve tasks desired by humans. Here our affordance model can be greatly beneficial by bootstrapping the exploration from the predicted affordances allowing the agent to focus on parts of the scene likely to be of interest to humans. To operationalize this, we first use the affordance model $f_\theta(.)$ for data-collection. We then rank all the trajectories collected using a task-agnostic exploration metric, and fit a distribution h to the (c, τ) values of the top trajectories. For subsequent data collection, we sample from h with some probability, and otherwise use the affordance model f. This process can then be repeated, and the elite-fitting scheme will bootstrap from highly exploratory trajectories to improve exploration even further. For the exploration metric in our experiments, we maximize *environment change* $\text{EC}(I_i, I_j) = ||\phi(I_i) - \phi(I_j)||_2$, (similar to previous exploration approaches [19, 280]) between first and last images in the trajectory, where ϕ masks the robot and the loss is only taken on non-masked pixels.

C. Goal-Conditioned Learning: While exploring the environment can lead to interesting skills, consider a robot that already knows its goal. Using this knowledge (*e.g.* an image of the opened door), it supervise its policy search. Goal images are frequently used to specify rewards in RL [416, 127, 271, 286, 123, 8, 270, 464, 254]. Using our affordance model can expedite the process of solving goal-specified tasks. Similar to the exploration setting, we rank trajectories and fit a distribution h to the (c, τ) values of the top trajectories, but here the metric is to minimize distance to the goal image I_g. The metric used in our experiments is to minimize $\text{EC}(I_T, I_g)$, where I_T is the last image in the trajectory, or to minimize $||\psi(I_g) - \psi(I_T)||_2^2$, where ψ is a feature space. Akin to exploration, subsequent data collection involves sampling from h and the affordance model f.

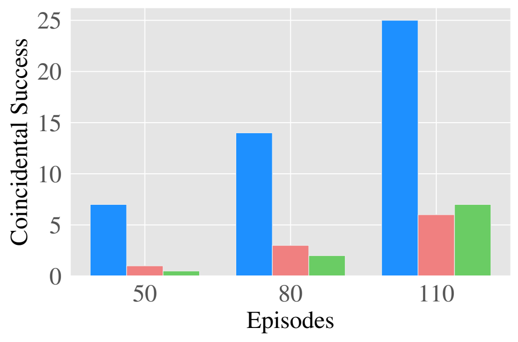
D. Affordance as an Action Space: Unlike games with discrete spaces like Chess and Go where reinforcement learning is deployed *tabula rasa*, robots need to operate in continuous action spaces that are difficult to optimize over. A pragmatic alternative to continuous action spaces is parameterizing them in a spatial manner and assigning a primitive (e.g. grasping, pushing or placing) to each location [454, 452, 364]. While this generally limits the type of tasks that can be performed, our affordance model already seeks out interesting states, due to the data it is trained on. We first query the affordance model on the scene many times to obtain a large number of predictions. We then fit a GMM to these points to obtain a discrete set of (c, τ) values, and now the robot just needs to search over this space.



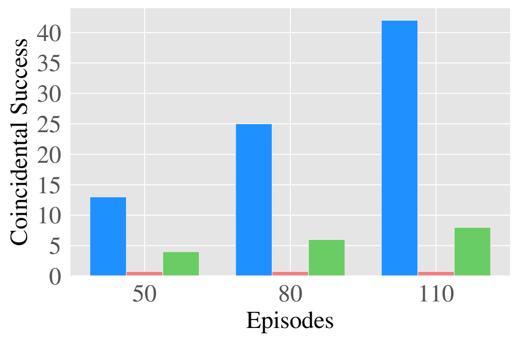
(a) Cabinet



(b) Knife



(c) Stovepot



(d) Shelf

Figure 4.5: **Exploration:** Coincidental success of VRB in comparison to random exploration or the exploration based on HAP [128].

4.4 Experimental Setup and Results

Through the four robot learning paradigms, shown in Fig. 4.3, we seek to answer the following questions: (1) Does our model enable a robot to collect *useful data* (imitation from offline data)?, (2) How much benefit does VRB provide to *exploration* methods?, (3) Can our method enable *goal-conditioned* learning?, and (4) Can our model be used to define a structured *action space* for robots? Finally, we also study whether our model learns meaningful *visual representations* for control as a byproduct and also analyze the *failure modes* and how they differ from prior work.

Robotics Setup: We use two different robot platforms - the Franka Emika Panda arm and the Hello Stretch mobile manipulator. We run the Franka on two distinct play kitchen environments and test on tasks that involve interacting with a cabinet, a knife and some vegetables, and manipulation of a a shelf and a pot. The Hello robot is tested on multiple in-the wild tasks outside lab settings, including opening a garbage can, lifting a lid, opening a door, pulling out a drawer, and opening a dishwasher (Fig. 4.1). We also provide support for a simulation environment on the Franka-Kitchen benchmark [114].

	Cabinet	**Knife**	**Veg**	**Shelf**	**Pot**	**Door**	**Lid**	**Drawer**
k-Nearest Neighbors:								
`HOI`	0.2	0.1	0.1	0.6	0.0	0.4	0.0	0.6
`HAP`	0.3	0.0	0.3	0.0	0.1	0.2	0.0	0.1
`Hotspots`	0.4	0.0	0.1	0.0	**0.5**	0.4	0.3	0.5
`Random`	0.3	0.0	0.1	0.3	0.4	0.2	0.1	0.2
`VRB` **(ours)**	**0.6**	**0.3**	**0.6**	**0.8**	0.4	**1.0**	**0.4**	**1.0**
Behavior Cloning:								
`HOI`	0.3	0.0	0.3	0.0	0.1	0.2	0.0	0.1
`HAP`	0.5	0.0	**0.4**	0.0	0.3	0.1	0.0	0.1
`Hotspots`	0.2	0.0	0.0	0.0	**0.8**	0.1	0.0	0.7
`Random`	0.1	**0.1**	0.1	0.0	0.2	0.1	0.0	0.0
`VRB` **(ours)**	**0.6**	**0.1**	0.3	**0.3**	**0.8**	**0.9**	**0.2**	**0.9**

Table 4.1: **Imitation Learning:** Success rate for k-NN and Behavior Cloning on collected offline data using various affordance models. We find that VRB vastly outperforms prior approaches, indicating better quality of data.

Observation and Action space: For each task, we estimate a task-space image-crop using bounding boxes [462], and pass random sub-crops to f_θ. The prediction for contact points c and post-contact trajectory τ is in pixel space, which are projected into 3D for robot control using a calibrated robot-camera system (with an Intel RealSense D415i). The robot operates in 6DOF end-effector space – samples a rotation, moves to a contact point, grasps, and then moves to a post-contact position (see Sec. 4.3.1).
Baselines and Ablations: We compare against prior work that has tried to predict heatmaps from human video : 1) Hotspots [267] 2) Hands as Probes (`HAP`) [128], a modified version for our robot setup of Liu *et al.* [236] that predicts contact region and forecast hand poses: 3) `HOI` [236] and 4) a baseline that samples affordances at random (`Random`). HAP and Hotspots only output a contact point, and we randomly select a post-contact direction.

4.4.1 Quality of Collected Data for Imitation

We investigate VRB as a tool for useful data collection. We evaluate this on both our robots across 8 different environments, with results in Tab. 4.1. These are all unseen scenarios (not in train set). Tasks are specified for each environment using goal images (eg - open door, lifted pot etc), and we use the data collected (30-150 episodes) for two established offline learning methods: (1) k-Nearest Neighbors (k-NN) and (2) Behavior Cloning. k-NN [279] finds trajectories in the dataset that are close (via distance in feature space [273]) to the goal image. We run the 10-closest trajectories to the goal image and record whether the robot has achieved the task specified in the goal image. For behavior cloning, we train a network supervised with (image, waypoint) pairs from the collected dataset, and the resulting policy is run 10 times on the real system. With both k-NN and BC, our method outperforms prior tasks on 7 out of 8 tasks, with an average success rate of 57 %, with the runner-up method (Hotspots [267]) only getting 25 %. This shows that VRB leads to much better data offline data quality, and thus can lead to better imitation learning performance. We additionally test for grasping held-out *rare* objects such as VR remotes or staplers, and find that VRB outperforms baselines.

4.4.2 Reward-Free Exploration

Here we study self-supervised exploration with no external rewards. We utilize environment change, *i.e.*, change in the position of objects as a task-agnostic metric for exploration [19]. For improved exploration, we bias sampling towards trajectories with a higher environment change metric. To evaluate the quality of exploration data, we measure how often does the robot achieves coincidental success *i.e.* reach a goal image configuration without having access to it. As shown in Fig. 4.5, we obtain consistent improvements over HAP [128] and random exploration raising performance multiple fold – from $3\times$ to $10\times$, for every task.

4.4.3 Goal-Conditioned Learning

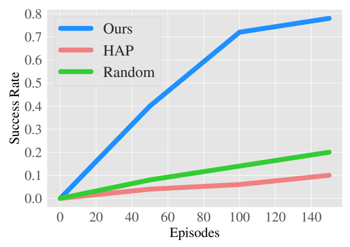
The previous settings help robots improve their behaviors with data without an external reward or goal. Here we focus on goal-driven robot learning. Goals are often specified through images of the goal configuration. Note that goal images are also used in Sec. 4.4.1 but as part of a static dataset to imitate. Here, the robot policy is updated with new data being added to the buffer. We sample this dataset for trajectories that minimize visual change with respect to the goal image. As shown in Fig. 4.6, VRB learns faster and better HAP [128] and Random on this robot learning paradigm, over six diverse tasks.

4.4.4 Affordance as an Action Space

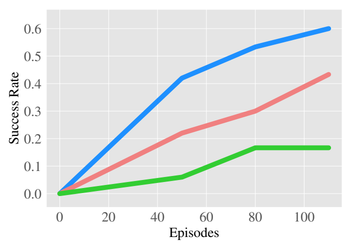
We utilize visual affordances to create a discrete action space using a set of contact points and interaction directions. We then train a Deep Q-Network (DQN) [258] over this action space, for the above goal-conditioned learning problem.In Fig. 4.7, we see that with VRB, the robot experiences more successes showing that a greater percentage of actions in the discretized action space correspond to meaningful object interactions.

	VRB	R3M [273]
microwave	**0.16**	0.10
slide-door	**0.84**	0.70
door-open	**0.13**	0.11

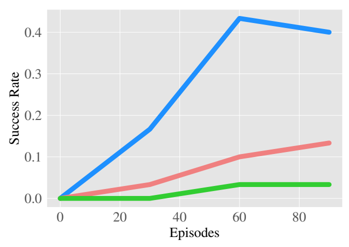
Table 4.2: Imitation with VRB vs. R3M [273] representation.



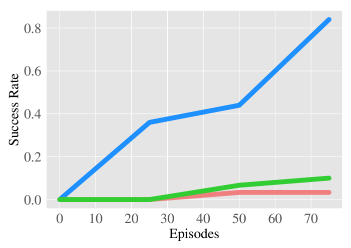
(a) Door



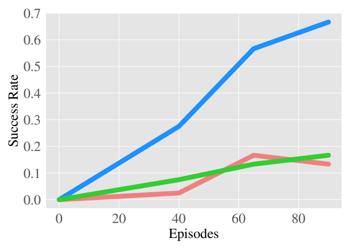
(b) Veggies



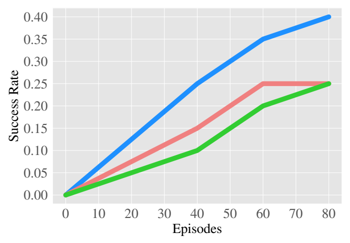
(c) Lid



(d) Dishwasher

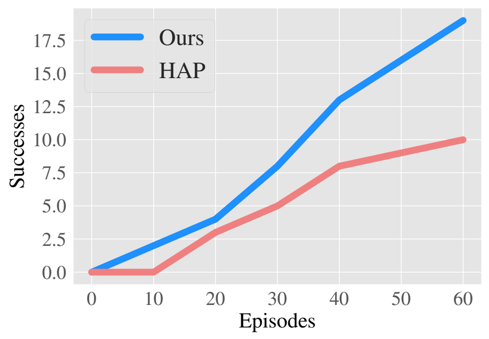


(e) Drawer

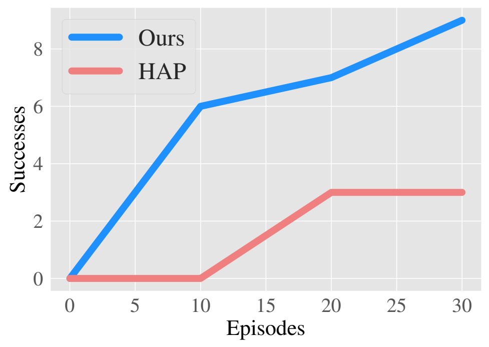


(f) Garbage Can

Figure 4.6: **Goal-conditioned Learning**: Success rate for reaching goal configuration for six different tasks. Sampling via VRB leads to faster learning and better final performance.



(a) Cabinet



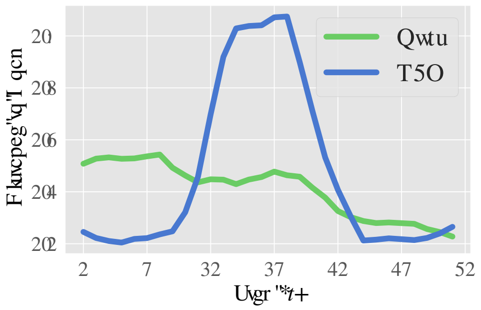
(b) Veggies

Figure 4.7: **Action Space**: Success using DQN with the discretized action space, for reaching a specified goal image.

4.4.5 Analyzing Visual Representations

Beyond showing better utility for robot learning paradigms, we analyze the quality of visual representations of the encoder learned in VRB. Two standard evaluations for this are (1) if they can help for downstream tasks and (2) how meaningful distances in their feature spaces are.

Fine-tuning: To investigate if the visual representations are effective for control, we directly fine-tune a policy on top of the (frozen) visual encoder. We evaluate on three simulated Franka environments, as shown in Tab. 4.2, and we see that VRB outperforms R3M on all tasks. We fine-tuned the policy only for 2K steps, instead of 20K in the R3M paper.



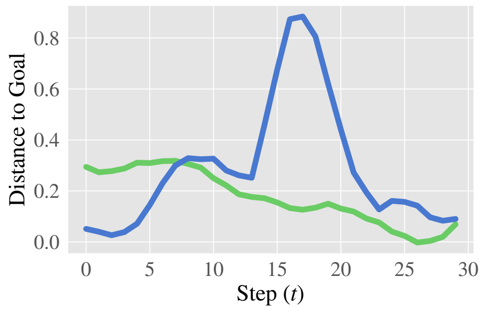


Figure 4.8: **Feature space distance**: Distance to goal in feature space for VRB decreases monotonically for door opening.

Feature space distance: We record the distance in feature space between the current and goal image for every timestep in the episode, for both VRB and R3M [273] on successful cabinet opening trajectories. As shown in Fig. 4.8, the distance for VRB decreases almost monotonically which correlates well with actual task progress.

4.4.6 Failure Modes

While VRB and the baselines see qualitatively similar successes, VRB in general sees a larger number of them and the *average case* scenario for VRB is much better. For the cabinet opening task, we classify each collected episode into three categories: "Failure", "Partial Success" and "Success". While VRB has a higher number of successful trajectories compared to the baselines (almost 2×), the number of partial successes is more than 6× (Fig. 4.9).

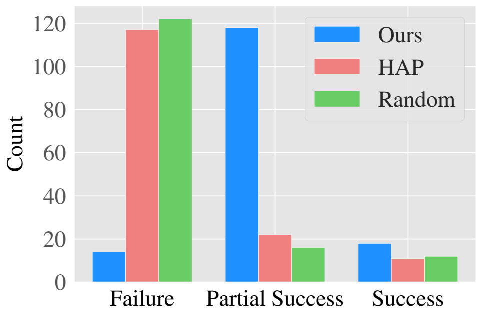


Figure 4.9: Failure mode analysis

4.5 Conclusion

We propose Vision-Robotics Bridge (VRB), a scalable approach for learning useful affordances from passive human video data and deploying them on many different robot learning paradigms (such as data collection for imitation, reward-free exploration, goal-conditioned learning, and parameterizing action spaces). Our affordance representation consists of contact points and post-contact trajectories. We demonstrate the effectiveness of this approach on the four paradigms and 10 different real-world robotics tasks, including many that are in the wild. We run thorough experiments, spanning over 200 hours, and show that VRB drastically outperforms prior approaches.

Chapter 5

Structured World Models from Human Videos

5.1 Motivation

A truly useful home robot should be general purpose, able to perform arbitrary manipulation tasks, and *get better* at performing new ones as it obtains more experience. How can we build such generalist agents? The current paradigm in robot learning is to train a policy, in simulation or directly in the real world, with engineered rewards or demonstrations directly constructed for the environment. While this has shown successes in lab-based tasks [182, 294, 221], learning is heavily dependent on the structure of the reward. This is not scalable as it is very challenging to *transfer* to new tasks, with different objectives. Often, it is also difficult to obtain ground truth objectives for a task in the real world. For a robot to succeed in the wild, it must not only learn many tasks at a time but also *get better* as it sees more data. How can we build an agent that can take advantage of large-scale experience and multi-task data?

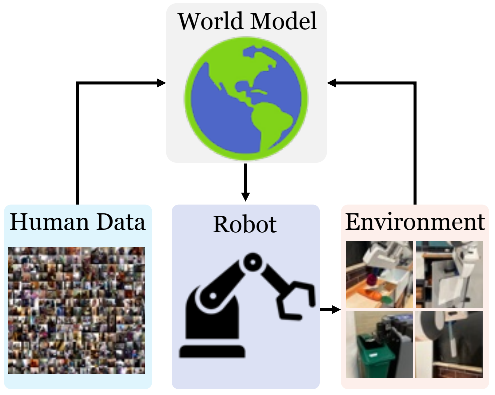


Figure 5.1: We present SWIM, an approach for learning manipulation tasks with only a handful of trajectories.

We aim to build *world models* to tackle this challenge. One key observation is that there is commonality between many tasks performed by humans on a daily basis. Even across diverse settings, the environment dynamics and physics share a similar structure. Learning a single *joint* world model, that predicts the future consequences of actions across diverse tasks can thus enable agents to extract this shared structure. While world models enable learning from inter-task data, they require action information to make predictions about the future. Furthermore, for planning in an

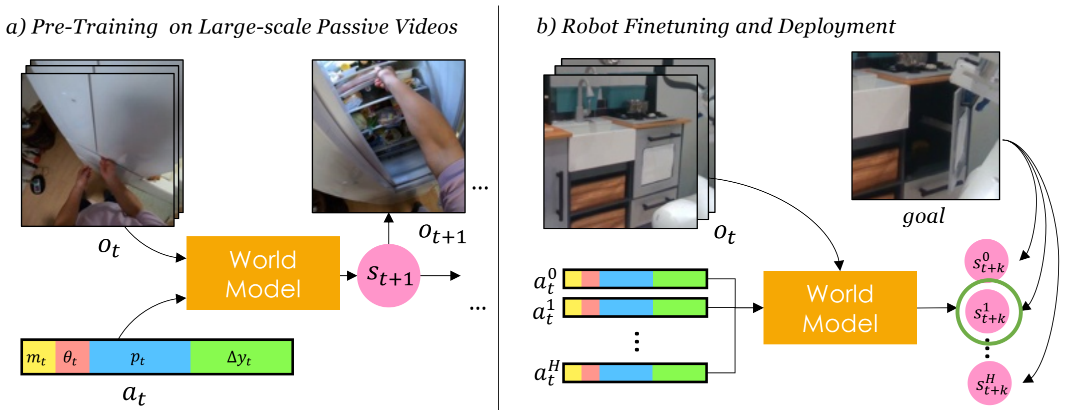

a) Pre-Training on Large-scale Passive Videos
o_t
World Model
s_{t+1}
o_{t+1}
m_t θ_t p_t Δy_t
a_t
b) Robot Finetuning and Deployment
o_t
goal
a_t^0
a_t^1
a_t^H
World Model
s_{t+k}^0
s_{t+k}^1
s_{t+k}^H

Figure 5.2: Overview of SWIM. We first pre-train the world model on a large set of human videos. We finetune this on many robot tasks, in an unsupervised manner, and deploy at test-time in the real world to achieve a given goal. Videos can be found at `https://human-world-model.github.io`

environment, the actions need to be relevant to the particular robot. Consequently, world models for robotics have mostly been trained only on data collected directly by a robot [97, 421, 216, 82, 98]. However, the quantity of this data is limited, which is very expensive and cumbersome to collect in the real world. Thus, the benefits of using large datasets as seen in other machine learning areas such as computer vision and language [40, 308] have not been realized for robotics, as no such dataset exists for robotics. However, there is an abundance of *human videos*, performing a very large set of tasks, on the internet. Is there a way to leverage this abundant data to learn world models for robotics, that will enable the robot to predict the consequences of its *actions* in any environment, enabling general-purpose learning? Due to the large morphology gap between robots and humans, it is challenging to obtain actions from human videos. Thus, previous approaches have mostly focused on learning visual representation features [273, 429] from observations alone. Using internet human videos to train robots requires us to define an action space that is applicable both in the human video domain and for robots. Consider the task of picking up a mug. To perform this task, the low-level signals sent to a person's arm compared to that of a robot would be completely different, and so predictive models in low-level joint space will not transfer well. If the action space instead required predicting the target pose and orientation of the mug handle, with low-level control abstracted away, then target poses used by humans could be utilized by robots as well. Thus, we learn *high-level* structured action spaces that are morphology invariant. For manipulation tasks, predicting a grasp location and post-grasp waypoints is an effective action space since it encourages object interaction. We can train visual affordance networks that produce these locations given videos leveraging techniques in computer vision [236, 128, 356, 267, 20].

We propose **S**tructured **W**orld **M**odels for **I**ntentionality (**SWIM**), which utilizes large-scale internet data to train world models for robotics using structured action spaces. Training the world model in the common high-level structured action space allows it to capture how human hands interact with objects when trying to grasp and manipulate them. This model can then be fine-tuned for robotics settings with only a handful of real-world interaction trajectories. This is because the world model can leverage the actionable representations it was pre-trained with due to the similarity in how the human hands from video data and robot grippers interact with the world. Furthermore, these interaction trajectories for fine-tuning do not require any task supervision and can be obtained simply by executing the visual affordance actions. We note that both pre-training on human videos and finetuning the world model on robot data do not make any assumption on rewards, and this *unsupervised* setting allows us to utilize data relevant for different tasks. This allows the robot to train a *single* world model on all the data, thus enabling us to train generalist agents. In our experiments, we show that we can train such joint world models through two distinct robot systems operating in real-world environments. Finally, we can deploy the fine-tuned world model to perform tasks of interest by specifying a goal image. The world model then plans in the affordance action space to find a sequence of actions to manipulate objects as required by the task.
To summarize, SWIM trains world models for robot control and consists of three stages: 1) Leveraging internet videos of human interactions for pre-training the model, 2) Finetuning the model to the robot setting using reward-free data, 3) Planning through the model to achieve goals. We evaluate this framework on two robot systems – a Franka Arm, and a Hello Stretch robot. SWIM is able to learn directly, is trained on data from multiple settings and gets better with data from more tasks. We perform a large-scale study across multiple environments and robots and find that SWIM achieves higher success ($\sim$ 2X) than prior approaches while being very sample efficient, requiring less than 30 minutes of real-world data.

5.2 Related Work

Efficient Real World Robot Learning: Deploying learning-driven approaches on hardware is challenging and requires either large engineering efforts to collect demonstrations [37, 176], many hours of autonomous interactions [182, 183], or simulations [9, 395, 206]. A major constraint of continuous control is the extremely large action space. Prior methods have focused on reducing this search space by using skills or options in a hierarchical manner [73, 284, 17, 387, 78, 78], physical inductive biases [261, 292, 202, 381, 199, 18, 248]. It is also possible to visually ground the action space, by parameterizing each observed location by a 2D [453, 452, 364, 175] or 3D [365] action. While these can speed up learning, our structured action space based on human-centric visual affordances allows us to not only perform manipulation efficiently but also leverage out-of-domain human/internet videos.

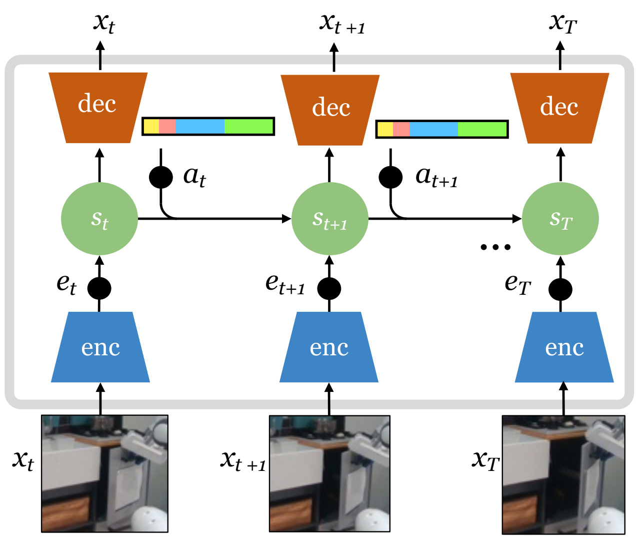


Figure 5.3: World Model Training: Images and actions are encoded into a learned feature space that has temporal structure, following the approach from [147]

Model-based learning: To tackle the sample efficiency problem in robot learning, prior methods have proposed learning dynamics models, which can later be used to optimize the policy [86, 87, 145, 70, 265, 266]. Such approaches mostly operate and learn in state space, which tends to be low dimensional. In order to deal with the highly complex visual observations from real-world settings, prior methods have used *World Models* [140], which capture dynamics of the *agent* and its *environment*. Such models can plan in image space [107, 97] or fully in imagination space [418, 146, 145, 217]. Such world models have been shown to be useful on a large set of tasks [321], including on hardware [425]. We argue that world models can be helpful in modeling the real world, especially if they can understand how the environment will behave at a high level and model the intentions of the agent.

Visual and Action Pre-Training for Robotics: In order to learn more generalizable and actionable representations, prior methods have learned visual encoders from large-scale human video data, either via video-language contrastive learning [273] or through inpainting masked patches [429, 309]. These representations have been shown to be useful for dynamics models as well [153]. Such approaches focus on the visual complexity of the world but do not encode any behavior information. Some works have incorporated low-level actions from human videos into the learning loop [244, 302, 12, 361], but these are fixed for a specific morphology and use a direct mapping to the robot. In contrast, our approach is able to learn a world model from human videos, incorporating action information, and works in multiple settings.

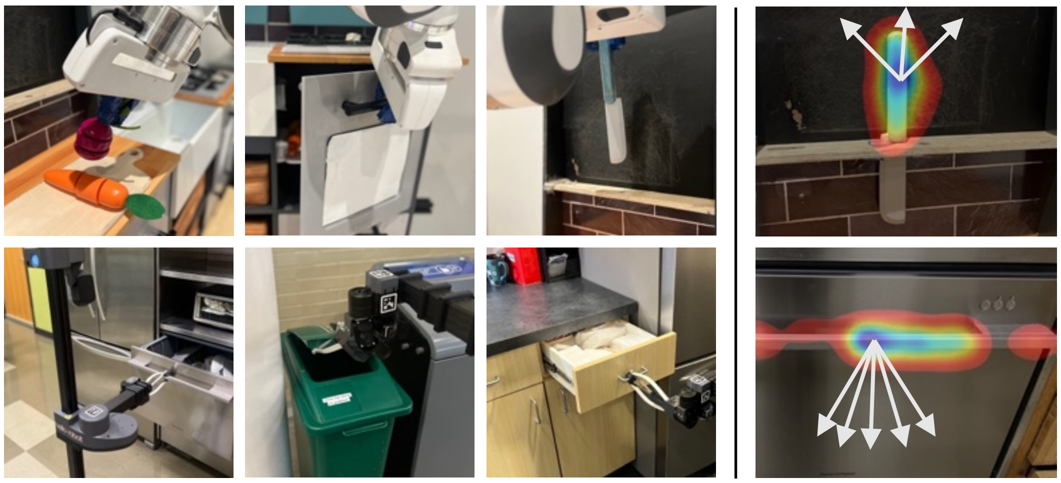


Figure 5.4: We evaluate SWIM on six different real-world manipulation tasks on two different robot systems (shown on the left). On the right, we show a sample of the visual affordances from the visual affordance model $\mathcal{G}_\psi$.

5.3 Background and Preliminaries

World Models: These are used to learn a compact state space for control given high-dimensional observations like images. The learned states preserve temporal information, which enables effective prediction and planning [140, 344, 343]. In this work, we use the model structure and training procedure from Dreamer [145, 146, 147], which has the following components:

$$\begin{aligned}
\text{encoder:} &\qquad e_t = \text{enc}_\phi(x_t) \\
\text{posterior:} &\qquad p(s_t|s_{t-1}, a_{t-1}, e_t) \\
\text{dynamics:} &\qquad p(s_t|s_{t-1}, a_{t-1}) \\
\text{decoders:} &\qquad p(x_t|s_t), p(r_t|s_t)
\end{aligned}$$

Here x_t, a_t, r_t denote the observation, action, and reward at time t, and s_t denotes the learned state space. Note that all these components are parameterized using neural networks. The model is trained by optimizing the ELBO as described in Dreamer, where the learned features are trained to reconstruct images and rewards and are regularized with a dynamics prior. The reward head decoder is not trained if r_t is not provided. For more details, we refer the readers to [147].

Hand-Object Interactions from Human Videos: In this work, leverage human videos to learn world models. Throughout the chapter, we will refer to a set of visual affordances. These visual affordances comprise of the hand trajectory h_t in image space (normalized to a 0-1 range), and object locations (o_t). We obtain human hand-object information (h_t, o_t) for each frame using the 100 Days of Hands [356] detector model, trained on many hours of youtube videos. These can then be used to identify where on the object the hand makes contact p^g, and we sample the hand position from a later frame in the video to obtain p^{pg}. Here p^g and p^{pg} denote the grasp and post-grasp pixel respectively and specify the visual affordance space.

5.4 World Models from Human Videos

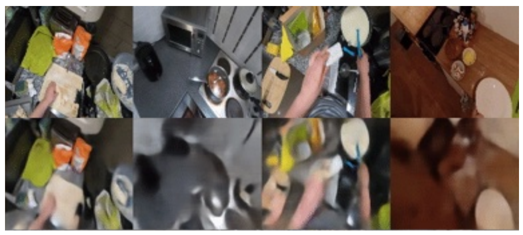
5.4.1 Visual Affordances as Actions

One of the key challenges is defining what the actions should be from human videos, most of which just contain image observations. Action information is essential for world models since they are required to learn dynamics and make predictions about the future. Furthermore, we need to define actions in a manner that is *transferable* from the human video domain to robot deployment settings. Following previous work that studies human-to-robot transfer for manipulation [19, 357, 56, 430, 370, 361, 360, 350, 451], we use the human hand motion in the videos to inform the action space. This is because we are focused on performing manipulation tasks, and how humans interact with objects using their hands contains useful information that can be transferred to robot end-effectors.

Structured Actions from Videos: We note that the videos of humans interacting with objects often consist of the hand moving to a point on the object, performing a grasp, and then manipulating the object. After obtaining the grasp pixel p^g and post-grasp pixels p^{pg}, using computer vision techniques similar to [128, 236, 267] from the video clip, we use these to train $\mathcal{G}_\phi$, which distills these labels into a neural network model conditioned on the first frame of the video clip. This model thus learns affordances associated with objects in the scene, by modeling how humans interact with them. This follows the affordances described in [20], but our work can also be combined with other affordance-learning approaches.

Transfer to Robot Scene: When dealing with 2D images, there is an inherent ambiguity regarding depth, which is required to map to a 3D point. To overcome this, we utilize depth camera observations to obtain the depth d_t^g at the image-space point p_t^g, and also sample the post-grasp depth d_t^{pg} within some range of the environment surface. This can then be projected into 3D coordinates in the robot frame, using hand-eye calibration, and the robot can attempt to grasp and manipulate objects by moving its gripper to these locations. The affordance action at time t can thus be expressed as $u_t = [p_t, d_t]$, where d_t is the depth corresponding to pixel p_t.

Hybrid Action Space: While visual affordances help structure the action space to increase the likelihood of useful manipulation and allow us to learn from human video, they impose restrictions on the full space of end-effector motion. Hence, we adopt a hybrid action space that has the option to execute both the aforementioned


(a) World Model pre-training reconstructions

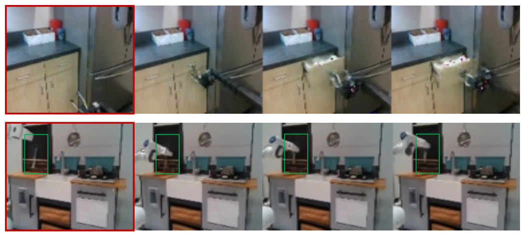

(b) Imagination rollouts at deployment

Figure 5.5: a) World Model pre-training reconstructions on Epic-Kitchens dataset [76]. b) Model imagination rollouts for high-reward trajectories. We can see that SWIM can imagine plausible and successful trajectories, for both human and robot data. The first image (highlighted in red) is the original observation by the robot.

visual affordance, as well as arbitrary end-effector Cartesian actions. We append a mode index to denote which type of action should be executed. This enables the robot to benefit both from the structured pixel-space visual affordance actions and the pre-training data in mode (m) 0, and make adjustments using arbitrary end-effector delta actions in mode 1. An action can be described by the following:

$$a_t = [m_t, \theta_t, u_t, \Delta y_t] \tag{5.1}$$

Here m_t denotes the mode, θ_t is the rotation of the gripper, u_t is the image-space action ($u_t = [p_t, d_t]$, where p_t are pixel coordinates in the image and d_t is depth), and Δy_t is the Cartesian end-effector action. At a particular timestep, only one out of the image action and Cartesian actions can be executed. If $m_t = 0$, this corresponds to the affordance mode, and so p_t is executed. If $m_t = 1$, then the robot is operating in the Cartesian control mode, and Δy_t is used. Due to our hybrid action space, we can seamlessly switch between training with the visual affordance and Cartesian end-effector action spaces. This allows the robot to leverage the structure from human video and also make adjustments if required using Cartesian actions which are useful for fine-grain control.

Algorithm 3 Human Video Data Training

```
Require: Human Video Dataset $\mathcal{D}$
1: initialize: World model $\mathcal{W}$, Affordance model $\mathcal{G}$
2: Process $\mathcal{D}$ into video clips $C^0, ...C^T$
3: Obtain grasp $p^g$ and post grasp $p^{pg}$ pixels for each $C^k$.
4: Create actions $a_t$ using eq. 5.1, with mode $m_t = 0$, and randomly sampling
   depth $d_t$ and rotation $\theta_t$
5: Train $\mathcal{G}_\phi(a^g, a^{pg}|I_0^k)$, where $I_0^k$ is the first frame of $C^k$
6: Train $\mathcal{W}$ on trajectory sequences $\{(I_0^k, a^g, I_{t1}^k, a^{pg}, I_{t2}^k)\}$
7: return $\mathcal{W}, \mathcal{G}$
```

	Cabinet	**Veg**	**Knife**	**Drawer**	**Dishwasher**	**Can**	**Average**
No world model:							
`BC-Affordance`	0.32	0.48	0.16	0.56	0.20	0.44	0.36
`BC-Pix`	0.16	0.40	0.00	0.24	0.08	0.12	0.17
No human-centric affordance-based actions:							
`MBRL-single` [147]	0.00	0.28	0.20	0.00	0.04	0.00	0.09
`MBRL-Pix-single`	0.52	0.36	0.16	0.00	0.04	0.04	0.19
No pre-training from human videos:							
`MBRL-Affordance-single`	0.68	0.16	0.40	0.84	0.20	0.36	0.44
`MBRL-Affordance-joint`	0.12	0.36	0.36	0.08	0.20	0.04	0.19
`SWIM`	0.84	0.76	**0.72**	0.92	**0.84**	**0.68**	**0.79**
`SWIM-single`	**0.88**	**0.80**	0.60	**0.96**	0.68	0.56	0.75

Table 5.1: Success rates of SWIM and baselines on 6 manipulation tasks, over 25 trials.

5.4.2 Structured Affordance-based World Models for Robotics

The overall approach is outlined in Alg. 4. We now describe each of the three phases - 1) World model pre-training on human videos, 2) Unsupervised finetuning with robot data, and 3) Robot deployment to perform a task given a goal image.

Training from Passive Human Videos: We first use a large set of human videos, obtained from Epic-Kitchens [76] to both train the world model $\mathcal{W}$, and obtain the visual affordance model $\mathcal{G}_\phi$. This dataset includes around 50k egocentric videos of people performing various manipulation tasks in kitchens. We first process this dataset into a set of short video clips (around 3 seconds). After obtaining the grasp pixel p^g and post-grasp pixels p^{pg} from the video clip, we convert them to our action space (specified in eq. 5.1), and train $\mathcal{G}_\phi$, as previously described in section 5.4.1. For video clip k, let I_t^k denote an image frame from the clip at time t. We collect images $I_{t_1}^k$ and $I_{t_2}^k$, where t_1 is the time of the grasp, and t_2 is when the hand is at p^{pg}. $\mathcal{W}$ is then trained on the trajectory sequences:

$$\{I_0^k, a^g I_{t_1}^k, a^{pg} I_{t_2}^k\} \tag{5.2}$$

This procedure is outlined in Alg. 3. As described in section 5.4.1, there are two modes for the actions - either in pixel space or end-effector space. In order to train on human videos, we consistently set $m_t = 0$ and thus use the image space actions. Since image depth and robot rotation information are not present in the video, we randomly sample values for these components. We include visualizations of the world model predictions on the passive data in Figure 5.5, and see that the model is able to capture the structure of the data.

Algorithm 4 Overview of SWIM

```
 1: Get $\mathcal{W}, \mathcal{G}$ = Human Video Data **Pre-Training** (Alg. 3)
 2: **Finetuning**: Query $\mathcal{G}$ for $N_0$ iterations to collect robot dataset $\mathcal{R_D}$ to train $\mathcal{W}$.
 3: **Task Deployment:** (Given goal $I_g$)
 4: Rank trajs in $\mathcal{R_D}$ using $I_g$. Fit GMM $g$.
 5: **for** traj 1:K **do**
 6:     Query $N$ proposals from $\mathcal{G}$, $\{a^g, a^{pg}\}_{1..N}$
 7:     Query $M$ proposals from $g$
 8:     Select best proposal using CEM through $\mathcal{W}$
 9:     Execute on the robot to reach $I_g$
10: **end for**
```

Finetuning with Robot Data: To use the world model $\mathcal{W}$ for control, we need to collect some in-domain robot data for finetuning. We do so by running the visual affordance model $\mathcal{G}$ to collect a robot dataset R_D, which is then used to train $\mathcal{W}$. We emphasize that this step does not require *any* supervision in the form of task rewards or goals. Hence, we can collect data from diverse tasks in the finetuning step. We see in Fig. 5.8 that SWIM enables the world model to pick up on the salient features of the robot environment very quickly as compared to models that do not use pre-training on human videos.

Task Deployment: After the world model has been fine-tuned on robot domain data, it can be used to perform tasks specified through goal images. The procedure for doing so is outlined in the Task Deployment section in Alg. 4. We collect two sets of action proposals. The first set is obtained by querying the visual affordance model $\mathcal{G}$ on the scene. We also want to leverage our knowledge of trajectories in $\mathcal{R_D}$ that reach states close to the goal. For this, we create a second set of proposals by fitting a Gaussian Mixture Model to the top trajectories in $\mathcal{R_D}$ and sampling from it. We then use the world model to optimize for an action sequence using the standard CEM approach [331], where the initial set of plans is set to be the combined set of action proposals. Ranking the trajectories in $\mathcal{R_D}$ and running CEM requires rewards, and we can obtain this by measuring the distance to the goal in the world model feature space:

$$r_t = \text{cosine}(f_{\mathcal{W}}(I_g), f_t)$$

where f_t is the world model feature, and $f_{\mathcal{W}}$ is the learned feature space of the model. For ranking trajectories in $\mathcal{R_D}$, $f_t = f_{\mathcal{W}}(I_k)$ for image k in the dataset. For planning, f_t corresponds to the predicted feature state. In our experiments, we use cosine distance to goal in the feature space from Nair et al. [273] to provide reward for model-free baselines, since they do not have a model, and so we also add this term to our reward by training a reward prediction head to get feature space [273] distance to goal from f_t. In our experiments we run an ablation where we use only the world model feature space, and find that performance for our approach is about the same.

5.5 Experimental Setup

5.5.1 Environments

Our real-world system consists of two different robots, evaluated over six tasks. Firstly, we use the Franka Emika arm, with end-effector control. This robot acts in a play kitchen environment with multiple tasks that mimic a real kitchen. Specifically, the robot needs to open a cabinet, pick up one of two toy vegetables from the counter and lift a knife from a holder. Note that the knife task is very challenging as it requires fine-grained control from the robot. In order to test SWIM in the wild we also deploy it on a mobile manipulator, the Stretch RE-1 from Hello-Robot. This is a collaborative robot designed with an axis-aligned set of joints and has suction cups as fingertips. We run this robot in real-world kitchens to perform different tasks, including opening a dishwasher, pulling out a drawer, and opening a garbage can. The garbage can task is challenging as the area for the robot to grasp onto is quite small. We show images of the environments in Figure 5.4.

5.5.2 Baselines and Ablations

In order to compare different aspects of SWIM, we run an extensive of baselines and ablations. All world-model-based approaches directly use code from Dreamer [147].

- `MBRL-Affordance:` An important contribution of SWIM is pre-training on human videos. This baseline is similar to SWIM but does not use any human video pre-training, allowing us to test our hypothesis that using human video is important for learning a generalizable world model.

- `MBRL-Pix:` Secondly, we would like to test how much the affordance action space helps the robot. This approach uses the same world model control procedure as SWIM, but does not sample actions using the visual affordance, $\mathcal{G}_\psi$. Instead grasp and post grasp locations are randomly sampled from an image crop around the object.

- `MBRL:` This baseline further removes structure from the action space, and only uses cartesian end-effector actions, without any pixel-space structure, thus $m_t = 1$ (described in section 5.4.1) for every timestep t. In order to help with sample efficiency, we use a simple heuristic to bootstrap this baseline: we initialize the robot each episode near the center of the detected object, using Detic [462], a state-of-the-art object detector.

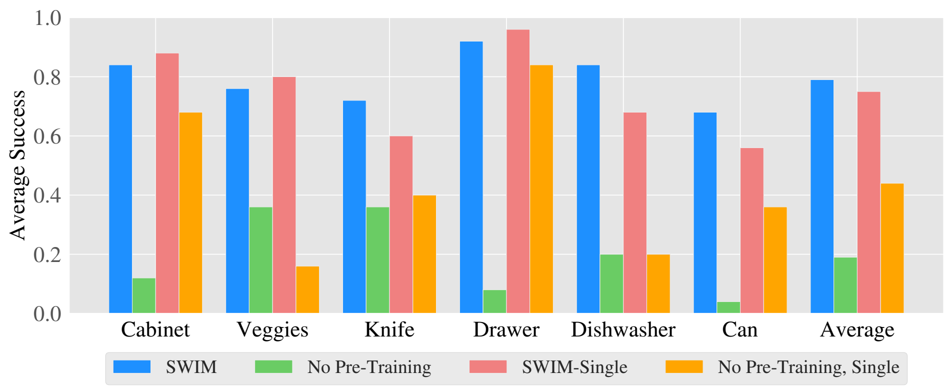


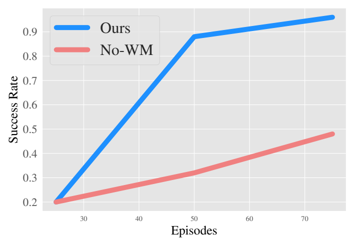
Figure 5.6: Comparison of SWIM and `MBRL-Affordance` for both the single task and jointly trained model. We see a large drop in success when removing pre-training on human videos, especially when dealing with diverse robot tasks.

- `BC-Affordance:` We would like to test if using world models is critical to performance, or if a simple behavior cloning approach can be effective. This baseline employs a filtered-behavior cloning [289, 291] strategy, in which the top trajectories based on reward (in our case distance to goal) are selected. Since there is no learned world model, we use distance in the feature space from the R3M model [273]. After selecting the top trajectories, we fit a gaussian mixture model and sample from it to obtain actions. These are in the same visual affordance action space used by our approach.

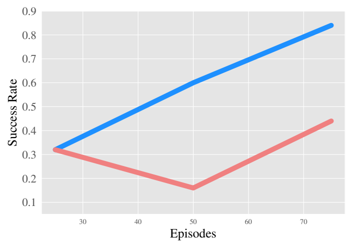
- `BC-Pix:` Uses behavior cloning in the same way as `BC-Affordance`. The only difference is the action space - this approach randomly samples locations and does not use $\mathcal{G}_\psi$ to obtain actions.

5.5.3 Implementation details

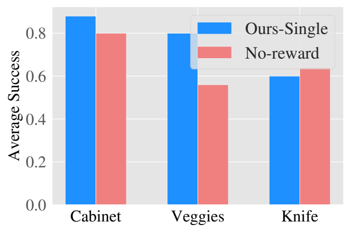
Human Video Data Pre-training: In order to pre-train the world model on human videos, we use the Epic-Kitchens [76] dataset. The dataset is divided into many small clips of humans performing semantic actions. We use the 100 Days of Hands [356] detector to find when an object has been grasped and find post grasp waypoints. Around 55K such clips are used to train the world model. Since we do not have depth or 3D information available, we randomly sample θ_t and the depth component of the image space action p_t.
Affordance Model: We show some qualitative examples of the affordances of the human-affordance model ($\mathcal{G}_\psi$) we use in Figure 5.4. This model has a UNet style encoder-decoder architecture, with a ResNet18 [157] encoder. The final output of the model is h_t and g_t, where g_t is a set of keypoints obtained from a spatial softmax



(a) Cabinet



(b) Knife



(c) No-Reward

Figure 5.7: Continuous improvement (a-b): SWIM continues to improve with online training, achieving high success. (c) Ablating the need for external feature space goal distance at test time.

over the network's heatmap outputs, representing the grasp point, and h_t is the post-grasp trajectory of the detected hand.

World Model: We use the world model from [147]. However, in order to handle high-dimensional image inputs we employ NVAE [400] as a stronger visual encoder. While not necessary to train the reward model q_r when finetuning, we empirically found that it added stability to the filtering setup a test time. We leave distilling the latent features into a neural distance function as future work.

Robot Deployment Setup: To capture videos and images we use an Intel Realsense D415, to get RGBD images. For each task we collect either 25 or 50 iterations of randomly sampled actions (in human-affordance, random image or Cartesian space), which takes about 30 minutes, finetuning the model on collected data. We obtain feature distance w.r.t. to image goals using the ResNet18 encoder from [273]. We sample around 2K action proposals and use the output of $\mathcal{W}_\phi$ to prune these. The model outputs are then evaluated (25 times). A human measures success based on a pre-defined metric (i.e. the cabinet should be fully open, etc).

5.6 Results

In our experiments we ask the following questions (i) Can we train a *single world model jointly* with data coming from diverse tasks? (ii) Does training the world model on *human video* data help performance? (iii) How important is our structured action space, based on human visual *affordances*? (iv) Are *world models* beneficial for learning manipulation with a handful of samples? (v) Can our approach *continually improve* performance with iterative finetuning?

We present a detailed quantitative analysis in Table 5.1. Across environment settings and robots SWIM achieves an average success rate of about 80% when using joint models (trained separately for Franka and Hello robot tasks). We also observe strong performance when SWIM is trained on individual tasks, getting an average success of 75%, compared to the next best approaches which only get around 40% success.

Joint World Model: A big benefit of SWIM is that it can deal with different sources of data. `SWIM-single` employs a model trained individually for each task. We see that overall the performance improves when sharing data, from the last two rows of Table 5.1. This is likely because there are some similarities across tasks that the model is able to capture. We find this encouraging and hope to scale to more tasks in the future. Further, we see that for the best baseline, `MBRL-Affordance-single`, using all the data jointly to train the world model leads to a major drop in performance (from 44% to 19%). We show this visually in the bar chart in Figure 5.6, where the effect of pre-training a model on human videos is amplified when dealing with all of the data from all the tasks. This shows that when dealing with a large set of tasks and diverse data, it is crucial to incorporate human-video pre-training for better performance and generalization. Hence we do not run joint world model experiments for `MBRL` and `MBRL-Pix` since the performance is already low (around 10 - 20 %) when the model is trained just on single task data.

Human Video Pre-Training: As noted in the previous section, pre-training on human videos is *critical* to being able to effectively train joint world models on multi-task data, as seen in Figure 5.6. For `MBRL-Affordance` we saw that in many cases the model collapses quickly to a sub-optimal control solution when trained on multiple tasks jointly. To investigate this further, we visualize the image reconstructions from W within the first minute of training and find that the outputs of SWIM were already very realistic, as compared to those of `MBRL-Affordance`. This can be seen in Figure 5.8, where the outputs of `MBRL-Affordance` are very pixelated while those of SWIM already capture important aspects of the ground truth, indicating the usefulness of pre-training on human videos.

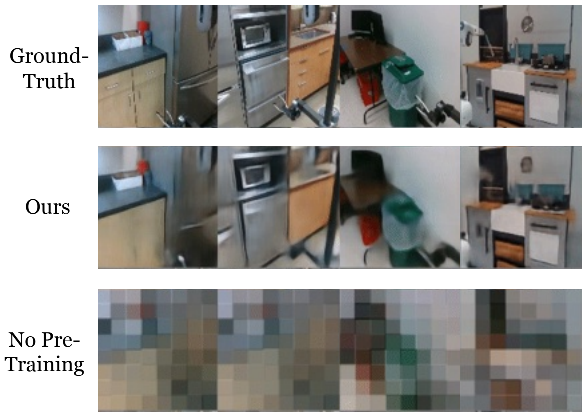


Figure 5.8: Image reconstruction using world model features in early training stages for SWIM and `MBRL-Affordance` (which has no pre-training), showing that SWIM can effectively transfer representations from human videos. Note that for our experiments we use models trained to convergence.

Human-Affordance Action Space: How does the choice of action space affect performance? For this we compare the (single task) model based and BC approaches separately. Comparing `MBRL-single` and `MBRL-Affordance-single` in Table 5.1, we can see that there is a clear benefit in using structured action spaces, with over 5X the success compared to cartesian end-effector actions. This fits our hypothesis, as it is very difficult for methods that use low-level actions to find successes in a relatively small number of interaction trajectories. The few successes that `MBRL-single` does see are due to the initialization of the robot close to the object using Detic. Furthermore, we note that this benefit is not simply because the affordance actions are in image space. In both the filtered-BC and world model case, the success rate with the affordance action space is roughly *double* than that of acting in pixel space, where target locations are sampled from a random crop around the object. This shows that picking the right action space *and* acting in a meaningful way to collect data can bootstrap learning and lead to efficient control.

Role of World Model: How important is using a world model, and can we achieve good performance by just using the affordance action space? From Table 5.1, we see that the average performance of `BC-Affordance` is not too far behind that of `MBRL-Affordance-single`. However, without a world model, the controller cannot leverage *multi-task* data, both for the pre-training stage to use human videos and for learning the shared structure across multiple robot domains by training a joint world model. Due to these critical reasons discussed previously, SWIM outperforms the best filtered-BC approach by more than a factor of 2.

Continual Improvement: Next, we investigate if SWIM can keep improving using the data that it collects when planning for the task. Since the world model can learn from all the data, we want to test if it can improve its proficiency on the task. Thus, after evaluating $\mathcal{W}$ once, we retrain on the newly collected data (as well as the old data), and re-evaluate the model. We present the learning curves in Figure 5.7. We see that SWIM is able to effectively improve performance, and achieves success of over 90%, which is far better than the performance of `BC-Affordance` even after continual training. This is an encouraging sign that SWIM can scale well since it can keep improving its performance with more data to continually learn. In the future, we hope to not only continually finetune, but also add new tasks and settings.

Reward Model: In Figure 5.7 c) we examine the effect of removing the reward prediction module on planning and find that only using distance in world model feature space is fairly competitive with using both the feature distance and predicted reward. We hypothesize that for the veggies task, it was harder to estimate reward accurately because the free objects tend to move around a lot during training, thus it might take more samples to learn consistent features.

5.7 Conclusion

In this chapter, we present SWIM, a simple and efficient way to perform many different complex robotics tasks with just a handful of trials. We aim to build a single model that can learn many tasks, as it holds the promise of being able to continuously learn and improve. We turn to a scalable source of useful data: human videos, from which we can model useful interactions. In order to overcome the morphology gap between robot and human videos, we create a structured action space based on human-centric affordances. This allows SWIM to pre-train a world model on human videos, after which it is fine-tuned using robot data collected in an unsupervised manner. The world model can then be deployed to solve manipulation tasks in the real world. The total robot interaction samples for the system can be collected in just 30 minutes. Videos of SWIM can be found at `https://human-world-model.github.io`. While SWIM provides a scalable solution and shows encouraging results, some limitations are in the types of actions and tasks that can be performed, as they currently only include quasi-static setups.

Chapter 6

Video Diffusion Alignment via Reward Gradients

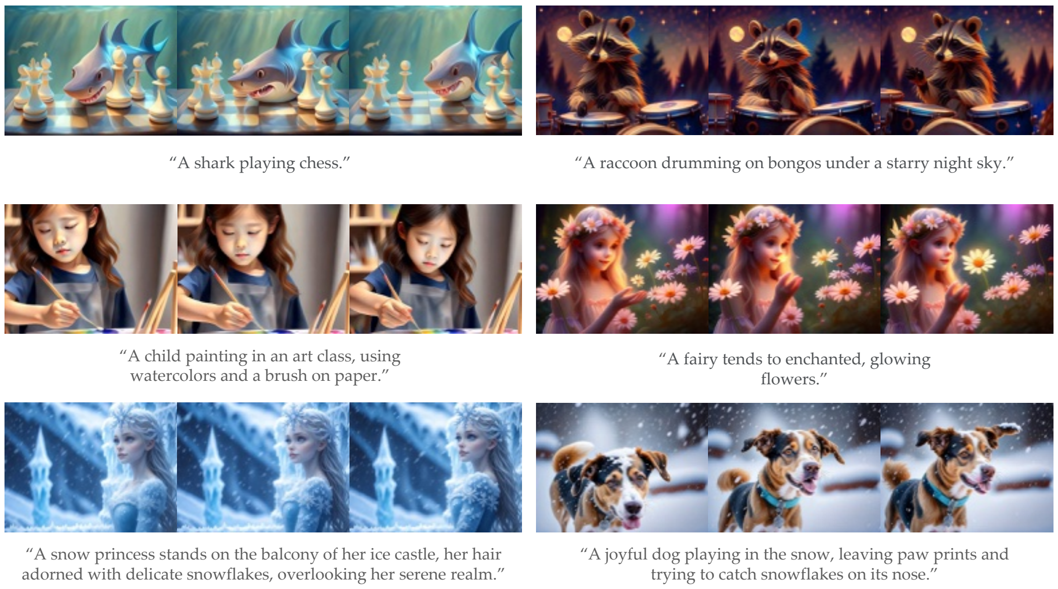

"A shark playing chess."
"A raccoon drumming on bongos under a starry night sky."
"A child painting in an art class, using watercolors and a brush on paper."
"A fairy tends to enchanted, glowing flowers."
"A snow princess stands on the balcony of her ice castle, her hair adorned with delicate snowflakes, overlooking her serene realm."
"A joyful dog playing in the snow, leaving paw prints and trying to catch snowflakes on its nose."

Figure 6.1: Generations from video diffusion models after adaptation with VADER using reward functions for aesthetics and text-image alignment. More visualization results are available at `https://vader-vid.github.io`

6.1 Motivation

We would like to build systems capable of generating videos for a wide array of applications, ranging from movie production, creative story-boarding, on-demand entertainment, AR/VR content generation, and planning for robotics. The most common current approach involves training foundational video diffusion models on extensive web-scale datasets. However, this strategy, while crucial, mainly produces videos that resemble typical online content, featuring dull colors, suboptimal camera angles, and inadequate alignment between text and video content.

Contrast this with the needs of an animator who wishes to bring a storyboard to life based on a script and a few preliminary sketches. Such creators are looking for output that not only adheres closely to the provided text but also maintains temporal consistency and showcases desirable camera perspectives. Relying on general-purpose generative models may not suffice to meet these specific requirements. This discrepancy stems from the fact that large-scale diffusion models are generally trained on a broad spectrum of internet videos, which does not guarantee their efficacy for particular applications. Training these models to maximize likelihood across a vast dataset does not necessarily translate to high-quality performance for specialized tasks. Moreover, the internet is a mixed bag when it comes to content quality, and models trained to maximize likelihood might inadvertently replicate lower-quality aspects of the data. This leads us to the question: How can we tailor diffusion models to produce videos that excel in task-specific objectives, ensuring they are well-aligned with the desired outcomes?

The conventional approach to aligning generative models in the language and image domains begins with supervised fine-tuning [312, 38]. This involves collecting a target dataset that contains expected behaviors, followed by fine-tuning the generative model on this dataset. Applying this strategy to video generation, however, presents a significantly greater challenge. It requires obtaining a dataset of target videos, a task that is not only more costly and laborious than similar endeavors in language or image domains, but also significantly more complex. Furthermore, even if we were able to collect a video target dataset, the process would have to be repeated for every new video task, making it prohibitively expensive. Is there a different source of signal we can use for aligning video diffusion, instead of trying to collect a target dataset of desired videos?

Reward models play a crucial role [345, 427, 211] in aligning image and text generations. These models are generally built on top of powerful image or text discriminative models such as CLIP or BERT [26, 308, 396]. To use them as reward models, people either fine-tune them via small amounts of human preferences data [345] or use them directly without any fine-tuning; for instance, CLIP can be used to improve image-text alignment or detectors can be used to remove objects in images [297].

This begs the question, how should reward models be used to adapt the generation pipeline of diffusion models? There are two broad categories of approaches, those that utilize reward gradients [297, 71, 433], and others that use the reward only as a scalar feedback and instead rely on estimated policy gradients [31, 219]. It has been previously found that utilizing the reward gradient directly to update the model can be much more efficient in terms of the number of reward queries, since the reward gradient contains rich information of how the reward function is affected by the diffusion generation [297, 71]. However, in text-to-image generation space, reward

gradient-free approaches are still dominant [336], since these can be easily trained within 24 hours and the efficiency gains of leveraging gradients are not significant. In this work, we find that as we increase the dimensionality of generation i.e transition from image to video, the gap between the reward gradient and policy gradient based approaches increases. This is because of the additional amount and increased specificity of feedback that is backpropagated to the model. For reward gradient based approaches, the feedback gradients linearly scale with respect to the generated resolution, as it yields distinct scalar feedback for each spatial and temporal dimension. In contrast, policy gradient methods receive a single scalar feedback for the entire video output. We test this hypothesis in Figure 6.6, where we find that the gap between reward gradient and policy gradient approaches increases as we increase the generated video resolution. We believe this makes it crucial to backpropagate reward gradient information for video diffusion alignment.
We propose VADER, an approach to adapt foundational video diffusion models using the gradients of reward models. VADER aligns various video diffusion models using a broad range of pre-trained vision models. Specifically, we show results of aligning text-to-video (VideoCrafter, OpenSora, and ModelScope) and image-to-video (Stable Video Diffusion) diffusion models, while using reward models that were trained on tasks such as image aesthetics, image-text alignment, object detection, video-action-classification, and video masked autoencoding. Further, we suggest various tricks to improve memory usage which allow us to train VADER on a single GPU with 16GB of VRAM. We include qualitative visualizations that show VADER significantly improves upon the base model generations across various tasks. We also show that VADER achieves much higher performance than alternative alignment methods that do not utilize reward gradients, such as DPO or DDPO. Finally, we show that alignment using VADER can easily generalize to prompts that were not seen during training. Our code is available at `https://vader-vid.github.io`.

6.2 Related Work

Denoising diffusion models [373, 160] have made significant progress in generative capabilities across various modalities such as images, videos and 3D shapes [161, 162, 235]. These models are trained using large-scale unsupervised or weakly supervised datasets. This form of training results in them having capabilities that are very general; however, most end use-cases of these models have specific requirements, such as high-fidelity generation [345] or better text alignment [427].
To be suitable for these use-cases, models are often fine-tuned using likelihood [32, 38] or reward-based objectives [31, 297, 71, 433, 219, 93, 106]. Likelihood objectives are often difficult to scale, as they require access to the preferred behaviour datasets. Reward or preference based datasets on the other hand are much easier to collect as they require a human to simply provide preference or reward for the data generated by the generative model. Further, widely available pre-trained vision

models can also be used as reward models, thus making it much easier to do reward fine-tuning [31, 297]. The standard approach for reward or preference based fine-tuning is to do reinforcement learning via policy gradients [31, 407]. For instance, the work of [219] does reward-weighted likelihood and the work of [31] applies PPO [347]. Recent works of [297, 71], find that instead of using policy gradients, directly backpropagating gradients from the reward model to diffusion process helps significantly with sample efficiency.
A recent method, DPO [312, 407], does not train an explicit reward model but instead directly optimizes on the human preference data. While this makes the pipeline much simpler, it doesn't solve the sample inefficiency issue of policy gradient methods, as it still backpropagates a single scalar feedback for the entire video output.
While we have made significant progress in aligning image diffusion models, this has remained challenging for video diffusion models [32, 411]. In this work, we take up this challenging task. We find that naively applying prior techniques of image alignment [297, 71] to video diffusion can result in significant memory overheads. Further, we demonstrate how widely available image or video discriminative models can be used to align video diffusion models. Concurrent to our work, InstructVideo [450] also aligns video diffusion models via human preference; however, this method requires access to a dataset of videos. Such a dataset is difficult to obtain for each different task, and becomes difficult to scale especially to large numbers of tasks. In this work, we show that one can easily align video diffusion models using pre-trained reward models while not assuming access to any video dataset.

6.3 Background

Diffusion models have emerged as a powerful paradigm in the field of generative modeling. These models operate by modeling a data distribution through a sequential process of adding and removing noise.
The forward diffusion process transforms a data sample x into a completely noised state over a series of steps T. This process is defined by the following equation:

$$x_t = \sqrt{\bar{\alpha}_t}x + \sqrt{1-\bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1}), \tag{6.1}$$

where ϵ represents noise drawn from a standard Gaussian distribution. Here, $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$ denotes the cumulative product of $\alpha_i = 1-\beta_i$, which indicates the proportion of the original data's signal retained at each timestep t.
The reverse diffusion process reconstructs the original data sample from its noised version by progressively denoising it through a learned model. This model is represented by $\epsilon_\theta(x_t; t)$ and estimates the noise ϵ added at each timestep t.
Diffusion models can easily be extended for conditional generation. This is achieved by adding c as an input to the denoising model:

$$\mathcal{L}_{\text{diff}}(\theta; \mathcal{D}') = \frac{1}{|\mathcal{D}'|} \sum_{x^i, c^i \in \mathcal{D}'} ||\epsilon_\theta(\sqrt{\bar{\alpha}_t} x^i + \sqrt{1 - \bar{\alpha}_t}\epsilon, c^i, t) - \epsilon||^2, \tag{6.2}$$

where $\mathcal{D}'$ denotes a dataset consisting of image-conditioning pairs. This loss function minimizes the distance between the estimated noise and the actual noise, and aligns with the variational lower bound for $\log p(x|c)$.
To sample from the learned distribution $p_\theta(x|c)$, one starts with a noise sample $x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ and iteratively applies the reverse diffusion process:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t, c) \right) + \sigma_t \mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}), \tag{6.3}$$

The above formulation captures the essence of diffusion models, which highlights their ability to generate structured data from random noise.

6.4 VADER: Video Diffusion via Reward Gradients

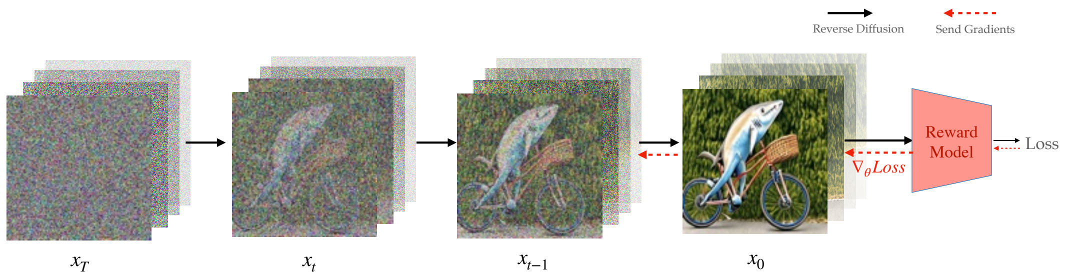


Figure 6.2: VADER aligns various pre-trained video diffusion models by backpropagating gradients from the reward model, to efficiently adapt to specific tasks.

We present our approach for adapting video diffusion models to perform a specific task specified via a reward function $R(.)$.
Given a video diffusion model $p_\theta(.)$, dataset of contexts D_c, and a reward function $R(.)$, we seek to maximize the following objective:

$$J(\theta) = \mathbb{E}_{c \sim D_c, x_0 \sim p_\theta(x_0|c)}[R(x_0, c)]$$

To learn efficiently, both in terms of the number of reward queries and compute time, we seek to utilize the gradient structure of the reward function, with respect to the weights θ of the diffusion model. This is applicable to all reward functions that are differentiable in nature. We compute the gradient $\nabla_\theta R(x_0, c)$ of these differentiable rewards, and use it to update the diffusion model weights θ. The gradient is :

$$\nabla_\theta R(x_0, c) = \sum_{t=0}^{T} \frac{\partial R(x_0, c)}{\partial x_t} \cdot \frac{\partial x_t}{\partial \theta}.$$

Algorithm 5 VADER

```
Require: Diffusion Model weights $\theta$
Require: Reward function $R(.)$
Require: Denoising Scheduler $f$
Require: Gradient cutoff step K
 1: while training do
 2:     for t = T,..1 do
 3:         pred = $\epsilon_\theta(x_t, c, t)$
 4:         if t ¿ K then
 5:             pred = stop_grad(pred)
 6:         end if
 7:         $x_{t-1}$ = $f$.step(pred, t, $x_t$)
 8:     end for
 9:     $g = \nabla_\theta R(x_0, c)$
10:     $\theta \leftarrow \theta - \eta * g$
11: end while
```

VADER is flexible in terms of the denoising schedule, we demonstrate results with DDIM [374] and EDM solver [186]. To prevent over-optimization, we utilize truncated backpropagation [389, 297, 71], where the gradient is back propagated only for K steps, where K ¡ T, where T is the total diffusion timesteps. Using a smaller value of K also reduces the memory burden of having to backpropagate gradients, making training more feasible. We provide the pseudocode of the full training process in Algorithm 5. Next, we discuss the type of reward functions we consider for aligning video models.

Reward Models: Consider a diffusion model that takes conditioning vector c as input and generates a video x_0 of length N, consisting of a series of images i_k, for each timestep k from 0 to N. Then the objective function we maximize is as follows:

$$J_\theta = \mathbb{E}_{c, i_{0:N}} \left[R([i_0, i_1 ... i_k ... i_{N-1}], c) \right]$$

We use a broad range of reward functions for aligning video diffusion models. Below we list down the distinct types of reward functions we consider.

Image-Text Similarity Reward: The generations from the diffusion model correspond to the text provided by the user as input. To ensure that the video is aligned with the text provided, we can define a reward that measures the similarity between the generated video and the provided text. To take advantage of popular, large-scale image-text models such as CLIP[308], we can take the following approach. For the entire video to be well aligned, each of the individual frames of the video likely need to have high similarity with the context c. Given an image-context similarity model g_{img}, we have:

$$R([i_0, i_1 ... i_k ... i_{N-1}], c) = \sum_k R(i_k, c) = \sum_k g_{\text{img}}(i_k, c)$$

Then, we have $J_\theta = \mathbb{E}_{k \in [0,N]} \left[g_{\text{img}}(i_k, c) \right]$, using linearity of expectation as in the image-alignment case. We conduct experiments using the HPS v2 [427] and PickScore [198] reward models for image-text alignment. As the above objective only sits on individual images, it could potentially result in a collapse, where the predicted images are the exact same or temporally incoherent. However, we don't find this to happen empirically, we think the initial pre-training sufficiently regularizes the fine-tuning process to prevent such cases.

Video-Text Similarity Reward: Instead of using per image similarity model g_{img}, it could be beneficial to evaluate the similarity between the whole video and the text. This would allow the model to generate videos where certain frames deviate from the context, allowing for richer, more diverse expressive generations. This also allows generating videos with more motion and movement, which is better captured by multiple frames. Given a video-text similarity model g_{vid} we have $J_\theta = \mathbb{E}\left[g_{\text{vid}}([i_0, i_1 ... i_k ... i_{N-1}], c)\right]$. In our experiments, we use a VideoMAE[396] fine-tuned on action classification, as g_{vid}, which can classify an input video into one of a set of action text descriptions. We provide the target class text as input to the text-to-video diffusion model, and use the predicted probability of the ground truth class from VideoMAE as the reward.

Image Generation Objective: While text similarity is a strong signal to optimize, some use cases might be better addressed by reward models that only sit on the generated image. There is a prevalence of powerful image-based discriminative models such as Object Detectors and Depth Predictors. These models utilize the image as input to produce various useful metrics of the image, which can be used as a reward. The generated video is likely to be better aligned with the task if the reward obtained on each of the generated frames is high. Hence we define the reward in this case to be the mean of the rewards evaluated on each of the individual frames, i.e $R([i_0, i_1 ... i_k ... i_{N-1}], c) = \sum_k R(i_k)$. Note that given the generated frames, this is independent of the text input c. Hence we have, $J_\theta = \mathbb{E}_{k \in [0,N]}\left[R(i_k)\right] = \mathbb{E}_{k \in [0,N]}\left[M_\phi(i_k)\right]$ via linearity of expectation, where M_ϕ is a discriminative model that takes an image as input to produce a metric, that can be used to define a reward. We use the Aesthetic Reward model [345] and Object Detector [104] reward model for our experiments.

Video Generation Objective: With access to an external model that takes in multiple image frames, we can directly optimize for desired qualities of the generated video. Given a video metric model N_ϕ, the corresponding reward is $J_\theta = \mathbb{E}\left[N_\phi([i_0, i_1, .. i_k ... i_{N-1}])\right]$.

Long-horizon consistent generation: In our experiments, we adopt this formulation to enable a feature that is quite challenging for many open-source video diffusion models - that of generating clips that are longer in length. For this task, we use Stable Video Diffusion [32], which is an image-to-video diffusion model. We increase the context length of Stable Video Diffusion by 3x by making it autoregressive. Specifically, we pass the last generated frame by the model as input for generating the next video sequence. However, we find this to not work well, as the model was never trained over its own generations thus resulting in a distribution shift. In order to improve the generations, we use a video metric model N_ϕ (V-JEPA [26]) that given a set of frames, produces a score about how predictive the frames are from one another. We apply this model on the autoregressive generations, to encourage

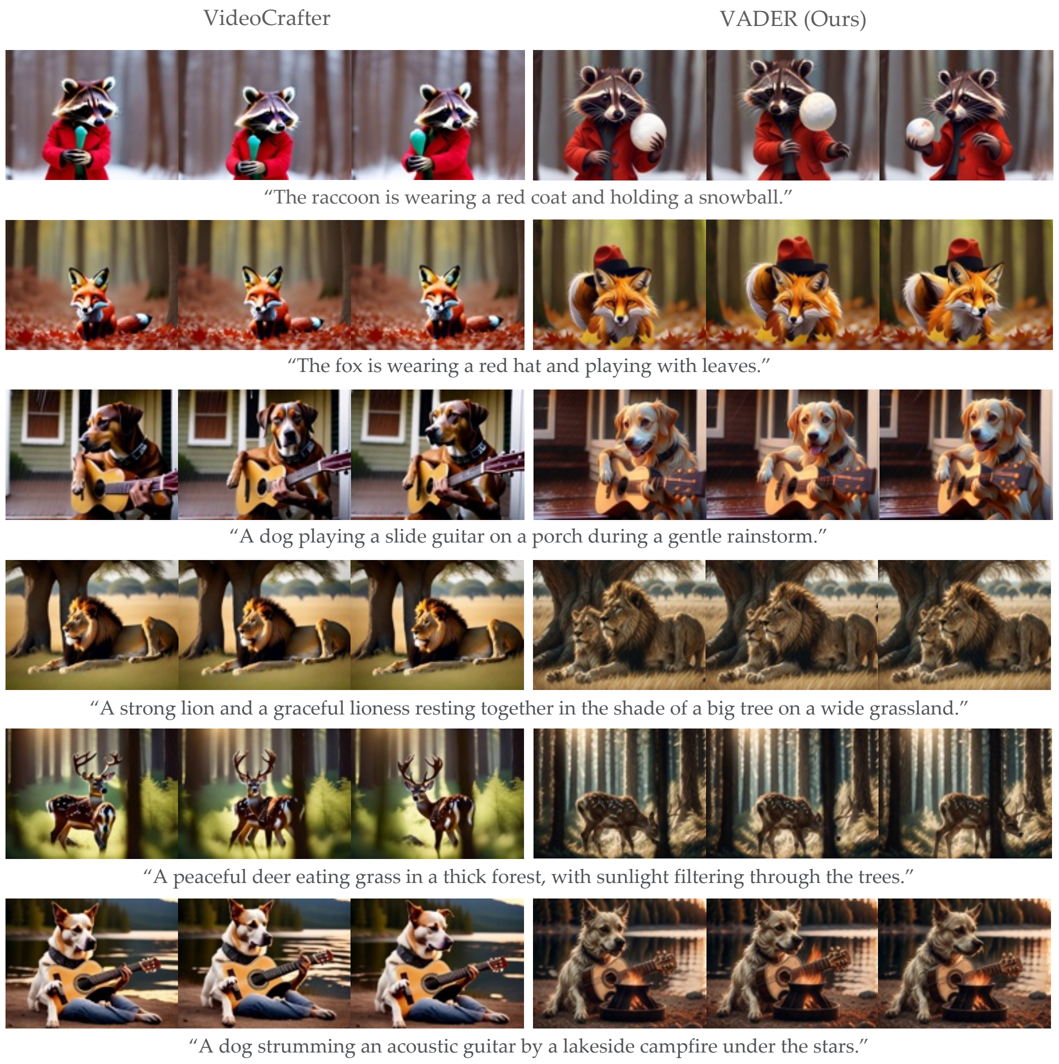

VideoCrafter
VADER (Ours)
"The raccoon is wearing a red coat and holding a snowball."
"The fox is wearing a red hat and playing with leaves."
"A dog playing a slide guitar on a porch during a gentle rainstorm."
"A strong lion and a graceful lioness resting together in the shade of a big tree on a wide grassland."
"A peaceful deer eating grass in a thick forest, with sunlight filtering through the trees."
"A dog strumming an acoustic guitar by a lakeside campfire under the stars."

Figure 6.3: Text-to-video generation results for VideoCrafter and VADER. We show results for VideoCrafter Text-to-Video model on the left and results for VADER on the right, where we use VideoCrafter as our base model. The reward models applied are a combination of HPSV2.1 and Aesthetic model in the first three rows, and PickScore in the last three rows. The videos in the third and last rows are generated based on prompts that are not encountered during training.

these to remain consistent with the earlier frames. Training the model in this manner allows us to make the video clips temporally and spatially coherent.

Reducing Memory Overhead: Training video diffusion models is very memory intensive, as the amount of memory linearly scales with respect to the number of generated frames. While VADER significantly improves the sample efficiency of

fine-tuning these models, it comes at the cost of increased memory. This is because the differentiable reward is computed on the generated frame, which is a result of sequential de-noising steps.
(i) **Standard Tricks:** To reduce the memory usage we use LoRA [164] that only updates a subset of the model parameters, further we use mixed precision that stores non-trainable parameters in fp16. To reduce memory usage during backpropagation we use gradient checkpointing and for the long horizon tasks, offload the storage of the backward computation graph from the GPU memory to the CPU memory.
(ii) **Truncated Backprop:** Additionally, In our experiments we only backpropagate through the diffusion model for one timestep, instead of backpropagating through multiple timesteps [297], and have found this approach to obtain competitive results while requiring much less memory.
(iii) **Subsampling Frames:** Since all the video diffusion models we consider are latent diffusion models, we further reduce memory usage by not decoding all the frames to RGB pixels. Instead, we randomly subsample the frames and only decode and apply loss on the subsampled ones. We conduct our experiments on 2 A6000 GPUS (48GB VRAM), and our model takes an average of 12 hours to train. However, our codebase supports training on a single GPU with 16GB VRAM.

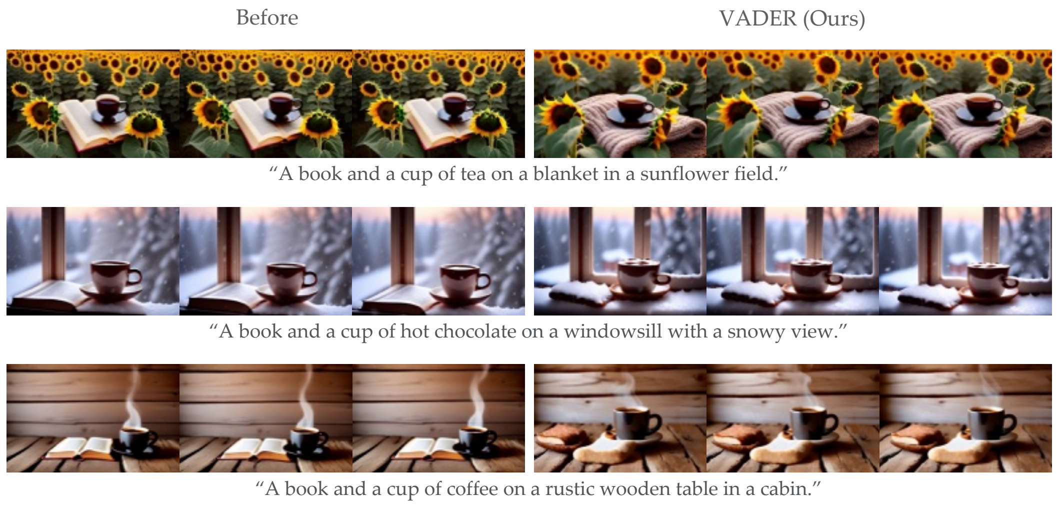

Before
VADER (Ours)
"A book and a cup of tea on a blanket in a sunflower field."
"A book and a cup of hot chocolate on a windowsill with a snowy view."
"A book and a cup of coffee on a rustic wooden table in a cabin."

Figure 6.4: Object removal using VADER. **Left:** Base model (VideoCrafter) generations, **Right:** VADER generations after fine-tuning to not display books using an object detector as a reward model. VADER effectively removes book and replaces it with some other object.

6.5 Results

In this work, we focus on fine-tuning various conditional video diffusion models, including VideoCrafter [57] , Open-Sora [461] , Stable Video Diffusion [32] and

ModelScope [411], through a comprehensive set of reward models tailored for images and videos. These include the Aesthetic model for images [345], HPSv2 [427] and PickScore [198] for image-text alignment, YOLOS [104] for object removal, VideoMAE for action classification [396], and V-JEPA [26] self-supervised loss for temporal consistency. Our experiments aim to answer the following questions:

- How does VADER compare against gradient-free techniques such as DDPO or DPO regarding sample efficiency and computational demand?
- To what extent can the model generalize to prompts that are not seen during training?
- How do the fine-tuned models compare against one another, as judged by human evaluators?
- How does VADER perform across a variety of image and video reward models?

This evaluation framework assesses the effectiveness of VADER in creating high-quality, aligned video content from a range of input conditioning.

Baselines: We compare VADER against the following methods:

- **VideoCrafter** [57], **Open-Sora 1.2** [461], and **ModelScope** [411] are current state-of-the-art (publicly available) text-to-video diffusion models. We serve them as base models for fine-tuning and comparison in our experiments in text-to-video space.
- **Stable Video Diffusion** [32] is the current state-of-art (publicly available) image-to-video diffusion model. In all our experiments in image-to-video space, we use their base model for fine-tuning and comparison.
- **DDPO** [31] is a recent image diffusion alignment method that uses policy gradients to adapt diffusion model weights. Specifically, it applies PPO algorithm [347] to the diffusion denoising process. We extend their code for adapting video diffusion models.
- **Diffusion-DPO** [407] extends the recent development of Direct Preference Optimization (DPO) [312] in the LLM space to image diffusion models. They show that directly modeling the likelihood using the preference data can alleviate the need for a reward model. We extend their implementation to aligning video diffusion models, where we use the reward model to obtain the required preference data.

Reward models: We use the following reward models to fine-tune the video diffusion model.

- **Aesthetic Reward Model**: We use the LAION aesthetic predictor V2 [345], which takes an image as input and outputs its aesthetic score in the range of 1-10. The model is trained on top of CLIP image embeddings, for which it uses a dataset of 176,000 image ratings provided by humans ranging from 1 to 10, where images rated as 10 are classified as art pieces.

- **Human Preference Reward Models**: We use HPSv2 [427] and PickScore [198], which take as input an image-text pair and predict human preference for the generated image. HPSv2 is trained by fine-tuning CLIP model with a vast dataset of 798,090 instances of human preference rankings among 433,760 image pairs, while PickScore [198] is trained by fine-tuning CLIP model with 584,000 examples of human preferences.

- **Object Removal**: We design a reward model based on YOLOS [104], a Vision Transformer based object detection model trained on 118,000 annotated images. Our reward is one minus the confidence score of the target object category, from which video models learns to remove the target object category from its video generation.

- **Video Action Classification**: While the above reward models sit on individual images, we employ a reward model that takes in the whole video as input. This can help with getting gradients for the temporal aspect of video generation. Specifically, we consider VideoMAE [396], which is fine-tuned for the task of action classification on Kinetics dataset [190]. Our reward is the probability predicted by the action classifier for the desired behavior.

- **Temporal Consistency via V-JEPA**: While action classification models are limited to a fixed set of action labels, here we consider a more general reward function. Specifically, we use self-supervised masked prediction objective as a reward function to improve temporal consistency. Specifically, we use V-JEPA [26] as our reward model, where the reward is the negative of the masked autoencoding loss in the V-JEPA feature space. Note that we employ exactly the same loss objective that V-JEPA uses in their training procedure.

Prompts: We consider the following set of prompt datasets for reward fine-tuning of text-to-video and image-to-video diffusion models.

- **Activity Prompts (Text):** We consider the activity prompts from the DDPO [31]. Each prompt is structured as ”a(n) [animal] [activity],” using a collection of 45 familiar animals. The activity for each prompt is selected from a trio of options: ”riding a bike”, ”playing chess”, and ”washing dishes”.

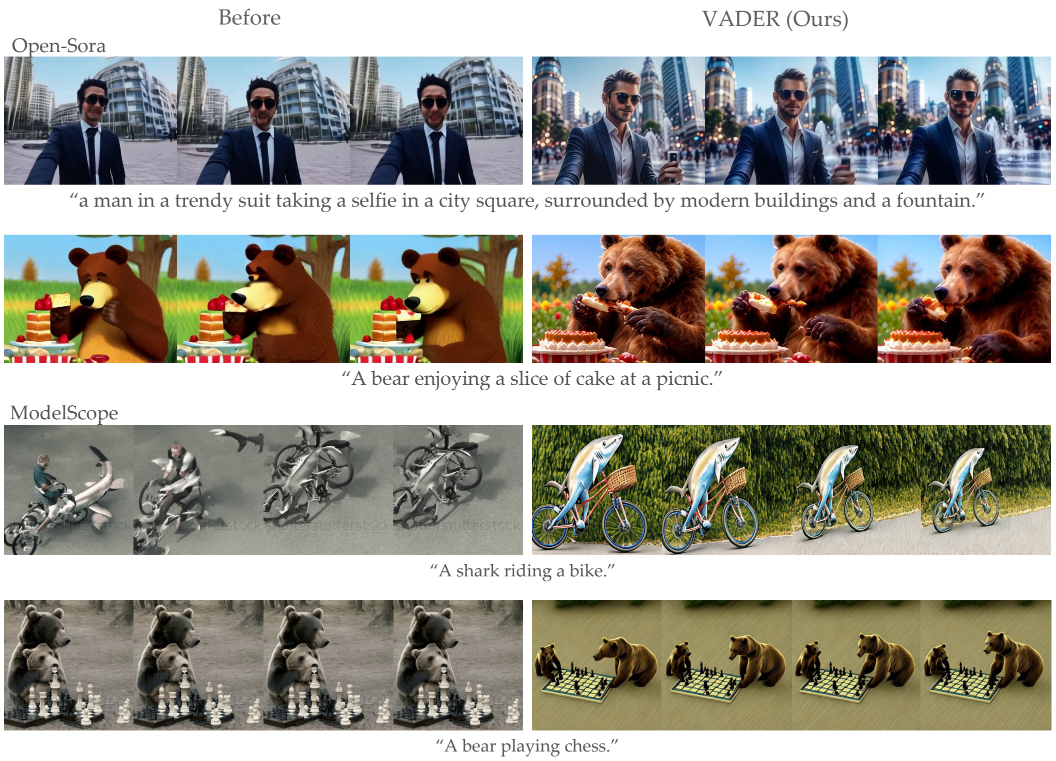

Before
VADER (Ours)
Open-Sora
"a man in a trendy suit taking a selfie in a city square, surrounded by modern buildings and a fountain."
"A bear enjoying a slice of cake at a picnic."
ModelScope
"A shark riding a bike."
"A bear playing chess."

Figure 6.5: Aligning Open-Sora 1.2 and ModelScope with VADER. The left column shows results from the base models, while results from VADER are demonstrated on the right. The first two rows use Open-Sora as the base model, and the last two rows use ModelScope. The reward models applied are PickScore in the first row, HPSv2.1 in the second row, HPSv2 in the third row, and the Aesthetic reward model in the last row.

- **HPSv2 Action Prompts (Text):** Here we filter out 50 prompts from a set of prompts introduced in the HPS v2 dataset for text-image alignment. We filter prompts such that they contain action or motion information in them.

- **ChatGPT Created Prompts (Text):** We prompt ChatGPT to generate some vivid and creatively designed text descriptions for various scenarios, such as books placed beside cups, animals dressed in clothing, and animals playing musical instruments.

- **ImageNet Dog Category (Image):** For image-to-video diffusion model, we consider the images in the Labrador retriever and Maltese category of ImageNet as our set of prompts.

- **Stable Diffusion Images (Image):** Here we consider all 25 images from Stable Diffusion online demo webpage.

6.5.1 Sample and Computational Efficiency

Training of large-scale video diffusion models is done by a small set of entities with access to a large amount of computing; however, fine-tuning of these models is done by a large set of entities with access to a small amount of computing. Thus, it becomes imperative to have fine-tuning approaches that boost both sample and computational efficiency.

In this section, we compare VADER's sample and computational efficiency with other reinforcement learning approaches such as DDPO and DPO. In Figure 6.7, we visualize the reward curves during training, where the x-axis in the upper half of the figure is the number of reward queries and the one in the bottom half is the GPU-hours. As can be seen, VADER is significantly more efficient in terms of sample and computation than DDPO or DPO. This is since we send dense gradients from the reward model to the diffusion weights, while the baselines only backpropagate scalar feedback.

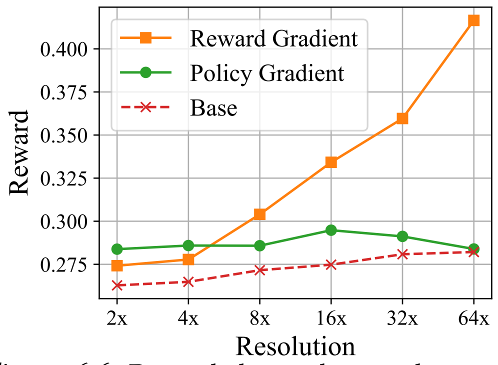


Figure 6.6: Reward obtained vs resolution of generated video for different methods. We report the reward achieved after 100 steps of optimization. As the resolution of the generation increases, the reward gap between VADER and DDPO significantly increases.

6.5.2 Generalization Ability

Method	**Aes (T2V)**		**HPS (T2V)**		**ActP**	**Aes (I2V)**	
	Train.	Test.	Train.	Test.	Train.	Train.	Test.
Base	4.61	4.49	0.25	0.24	0.14	4.91	4.96
DDPO	4.63	4.52	0.24	0.23	0.21	N/A	N/A
DPO	4.71	4.41	0.25	0.24	0.23	N/A	N/A
Ours	**7.31**	**7.12**	**0.33**	**0.32**	**0.79**	**7.83**	**7.64**

Table 6.1: Reward on Prompts in train & test. We split the prompts into train and test sets, such that the prompts in the test set do not have any overlap with the ones for training. We find that VADER achieves the best on both metrics.

A desired property of fine-tuning is generalization, i.e. the model fine-tuned on a limited set of prompts has the ability to generalize to unseen prompts. In this section, we extensively evaluate this property across multiple reward models and baselines. While training text-to-video (T2V) models, we use HPSv2 Action Prompts in our

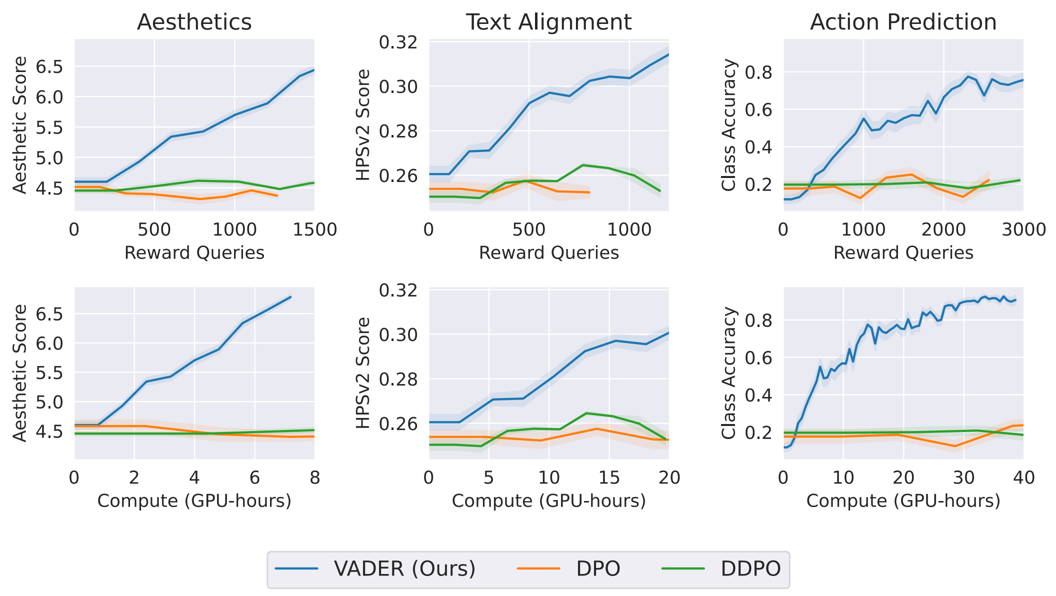


Figure 6.7: Training efficiency comparison. **Top**: Sample efficiency comparison with DPO and DDPO. **Bottom**: Computational efficiency comparison with DPO and DDPO. It can be seen that VADER starts converging within at most 12 GPU-hours of training, while DPO or DDPO do not show much improvement.

training set, whereas we use Activity Prompts in our test set. We use Labrador dog category in our training set for training image-to-video (I2V) models, while Maltese category forms our test set. Table 6.1 showcases VADER's generalization ability.

6.5.3 Human Evaluation

We carried out a study to evaluate human preferences via Amazon Mechanical Turk. The test consisted of a side-by-side comparison between VADER and ModelScope. To test how well the videos sampled from both the models aligned with their text prompts, we showed participants two videos generated by both VADER and a baseline method, asking them to identify which video better matched the given text. For evaluating video quality, we asked participants to compare two videos generated in response to the same prompt, one from VADER and one from a baseline, and decide which video's quality seemed higher. We gathered 100 responses for each comparison. The results, illustrated in Table 6.2, show a preference for VADER over the baseline methods.

Method	Fidelity	Text Align
ModelScope	21.0%	39.0%
VADER (**Ours**)	**79.0%**	**61.0%**

Table 6.2: Human Evaluation results for HPS reward model, where the task is image-text alignment.

6.5.4 Qualitative Visualization

In this section, we visualize the generated videos for VADER and the respective baseline. We conduct extensive visualizations across all the considered reward functions on various base models.

HPS Reward Model: In Figure 6.3, we visualize the results before and after fine-tuning VideoCrafter using both HPSv2.1 and Aesthetic reward function together in the top three rows. Before fine-tuning, the raccoon does not hold a snowball, and the fox wears no hat, which is not aligned with the text description; however, the videos generated from VADER does not result in these inconsistencies. Further, VADER successfully generalizes to unseen prompts as shown in the third row of Figure 6.3, where the dog's paw is less like a human hand than the video on the left. Similar improvements can be observed in videos generated from Open-Sora V1.2 and ModelScope as shown in the second and third rows of Figure 6.5.

Aesthetic Reward Model: In Figure 6.3, in the top three rows we visualize the results before and after fine-tuning ModelScope using a combination of Aesthetic reward function and HPSv2.1 model. Also, we fine-tune ModelScope via Aesthetic Reward function and demonstrate its generated video in the last row in Figure 6.5. We observe that Aesthetic fine-tuning makes the generated videos more artistic.

PickScore Model: In the bottom three rows of Figure 6.3, we showcase videos generated by PickScore fine-tuned VideoCrafter. VADER shows improved text-video alignment than the base model. In the last row, we test both models using a prompt that is not seen during training time. Further, video generated from PickScore fine-tuned Open-Sora is displayed in the first row of Figure 6.5.

Object Removal: Figure 6.4 displays the videos generated by VideoCrafter after fine-tuning using the YOLOS-based objection removal reward function. In this example, books are the target objects for removal. These videos demonstrate the successful replacement of books with alternative objects, like a blanket or bread.

Video Action Classification: In Figure 6.8, we visualize the video generation of ModelScope and VADER. In this case, we fine-tune VADER using the action classification objective, for the action specified in the prompt. For the prompt, "A person eating donuts", we find that VADER makes the human face more evident along with adding sprinkles to the donut. Earlier generations are often misclassified as baking cookies, which is a different action class in the kinetics dataset. The addition of colors and sprinkles to the donut makes it more distinguishable from cookies leading to a higher reward.

“A person eating Donuts”

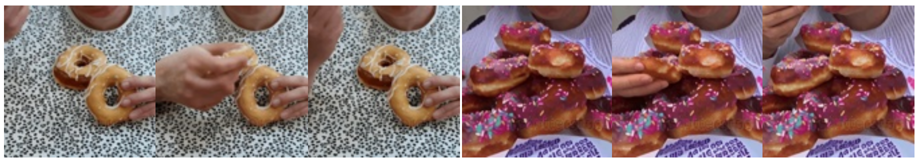


Figure 6.8: Video action classifiers as reward model. We use VideoMAE action classification model as a reward function to fine-tune ModelScope's Text-to-Video Model. We see that after fine-tuning, VADER generates videos that correspond better to the actions.

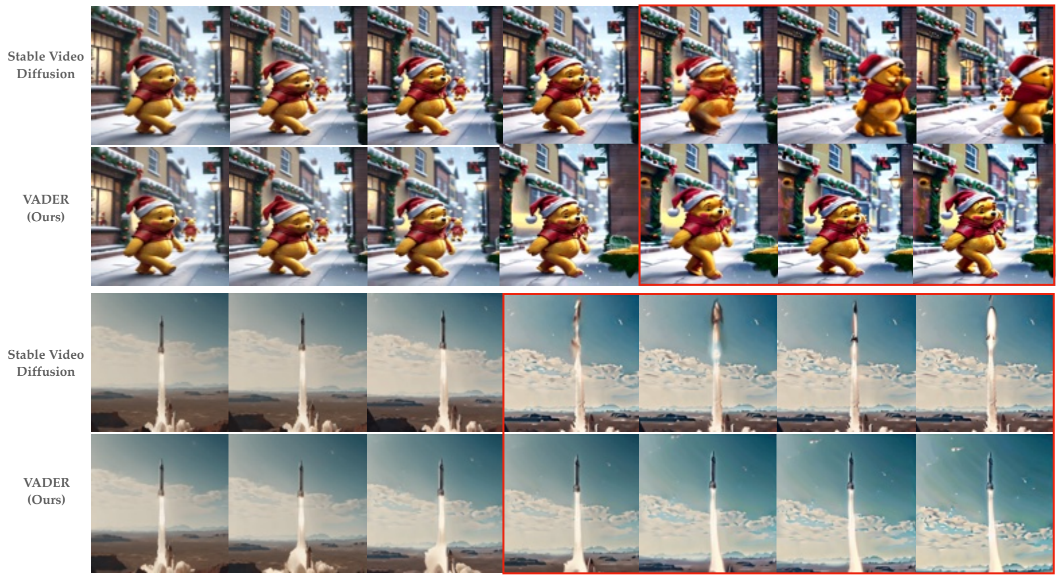


Figure 6.9: Improving temporal and spatial consistency of Stable Video Diffusion (SVD) Image-to-Video Model. Given the leftmost frame as input, we use autoregressive inference to generate 3*N frames in the future, where N is the context length of SVD. However, this suffers from error accumulation, resulting in corrupted frames, as highlighted in the red border. We find that VADER can improve the spatio-temporal consistency of SVD by using V-JEPA's masked encoding loss as its reward function.

V-JEPA reward model: In Figure 6.9, we show results for increasing the length of the video generated by Stable Video Diffusion (SVD). For generating long-range videos on SVD, we use autoregressive inference, where the last frame generated by SVD is given as conditioning input for generating the next set of images. We perform three steps of inference, thus expanding the context length of SVD by three times. However, as one can see in the images bordered in red, after one step of inference, SVD starts accumulating errors in its predictions. This results in deforming the teddy bear, or affecting the rocket in motion. VADER uses V-JEPA objective of masked encoding to enforce self-consistency in the generated video. This manages to resolve the temporal and spatial discrepancy in the generations as shown in Figure 6.9.

6.6 Conclusion

We presented VADER, which is a sample and compute efficient framework for fine-tuning pre-trained video diffusion models via reward gradients. We utilized various reward functions evaluated on images or videos to fine-tune the video diffusion model. We further showcased that our framework is agnostic to conditioning and can work on both text-to-video and image-to-video diffusion models. We hope our work creates more interest towards adapting video diffusion models.

Part III

Autonomous Exploration for Mobile Manipulation

Chapter 7

Continuously Improving Mobile Manipulation with Autonomous Real-World RL

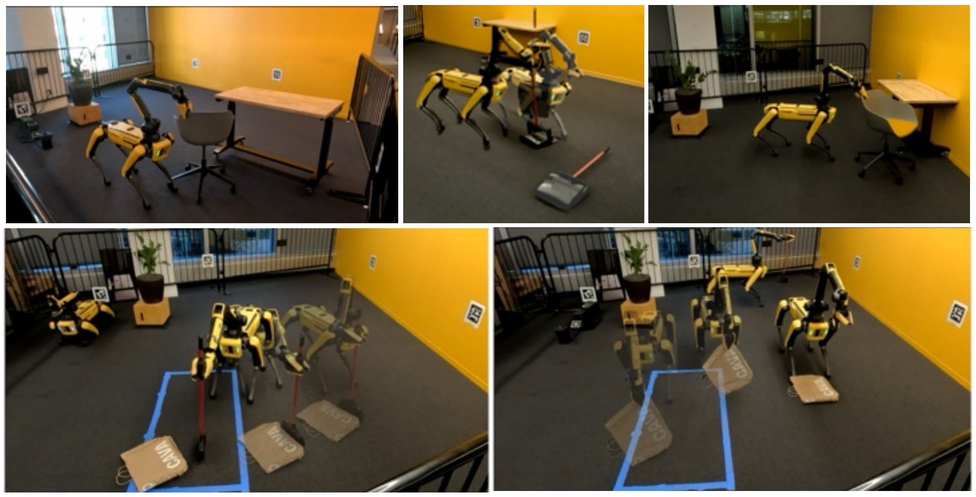


Figure 7.1: **Real World Autonomous Learning:** We enable a legged mobile manipulator to learn a variety of tasks such as moving chairs (top, left and right), righting a dustpan (top, middle), and sweeping (bottom) via autonomous practice in the real world.

7.1 Motivation

How do we build generalist systems capable of executing a wide array of tasks across diverse environments, with minimal human involvement? While visuomotor policies trained with reinforcement learning (RL) have demonstrated significant potential to bring robots into open-world environments, they often first require training in simulation [72, 394, 6, 144, 439, 61]. However, it is challenging to build simulations that capture the unbounded diversity of real-life tasks, especially involving complex manipulation. What if learning instead occurs through direct engagement with the real world, without extensive environment instrumentation or human supervision? Prior work on real-world RL for learning new skills has been shown for locomotion [371, 425], and in manipulation for pick-place [182, 183, 383, 159] or dexterous in-hand tasks [135, 434, 266] in stationary setups. Consider a complex, high-dimensional system like a legged mobile manipulator learning in open spaces. The feasible space of exploration is much larger than in constrained tabletop setups. **Autonomous operation** of such a complex, high-dimensional robots often does not result in data that has useful learning signal. For example, we would like to avoid the robot simply waving its arm in the air without interacting with objects. Furthermore, even after making some progress on the task, the robot should not stagnate near goal states. While prior work has explored using goal cycles [150, 135, 136] to help maintain state diversity, this has not been shown for mobile systems. Such systems also need to learn more complex skills, involving constrained manipulation of larger objects and moving beyond pick and place, making **sample-efficient learning** critical. Finally, **reward supervision** using current RL approaches often requires physical instrumentation using specialized sensors [437, 342] or humans in the loop [113, 368, 231, 370], which is difficult to scale to different tasks.

Our approach tackles each of these issues of autonomy, efficient policy learning, and reward specification. We enable higher-quality data collection by guiding exploration toward object interactions using off-the-shelf visual models. This leads the robot to search for, navigate to, and grasp objects before learning how to manipulate them. We preserve state diversity to prevent robot stagnation by extending the approach of goal-cycles to mobile manipulation tasks. For sample efficient policy learning, we combine RL with *behavior priors* that contain basic task knowledge. These priors can be planners with a simplified incomplete model, or procedurally generated motions. For rewards without instrumentation or human involvement, we combine semantic information from detection and segmentation models with low-level depth observations for object state estimation.

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 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 success rate of about 80% across tasks, a **4$\times$ improvement** over using either RL or the behavior prior individually with our task-relevant autonomy component.

7.2 Related Work

Autonomous Real-World RL: Previous work for real-world RL mostly involves either manipulation for table-top pick-place settings [182, 183, 425], in-hand dexterous manipulation [266, 135, 434] or locomotion behavior [142, 372, 425]. Approaches for automated resets needed for continual practice include instrumented environments [182, 183], forward-backward policies [359], graph structure of sub-tasks that serve as resets for one another [135, 434], or pre-trained, reliable reset policies [371]. For mobile manipulation, real-world RL has been limited to pick and place tasks [383, 159, 177], or operating small cabinets/ drawers [163]. There is also interesting work in using the legs of quadrupeds as manipulators [11, 60], but this is limited by the restricted degree of possible manipulation. In our work, we extend the RL framework to learn challenging manipulation skills such as sweeping and moving chairs for a mobile system. For efficient learning on these complex tasks, we leverage behavior priors, which have some basic task knowledge. Moreover, task specification is a big challenge [463] for real-world learning. Current approaches often require physical instrumentation using specialized sensors [437, 342] or humans in the loop [113, 368, 231, 370], which is difficult to scale to different tasks. There has been some work on completely self-supervised learning systems with some extensions to robotics [295, 251], but these approaches are challenging to deploy on complex tasks due to intractability, underspecification, and misalignment. We extend the approach of using language goals and combining these with large-scale visual models [197], conditioned on open-vocabulary prediction [449, 222, 27], to obtain object states, which can be used to compute reward.

Mobile Manipulation: 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 [204]. 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 [424, 422, 383, 21]. Imitation learning approaches to mobile manipulation enable joint reasoning over manipulation and navigation actions and generalize across broad sets of tasks [133, 37, 5, 353, 116]. However, imitation learning requires an expensive collection of expert trajectories. In contrast, RL methods can learn from experience without requiring extra human labor for each new task. Decomposing the action space over which the RL policy operates enables more tractable and efficient learning of long-horizon mobile manipulation skills [428, 131, 242, 445]. In our work, we move beyond tasks that involve picking and placing to instead learn skills that require coordination between the legs and arms, e.g., moving chairs or sweeping paper bags with a broom.

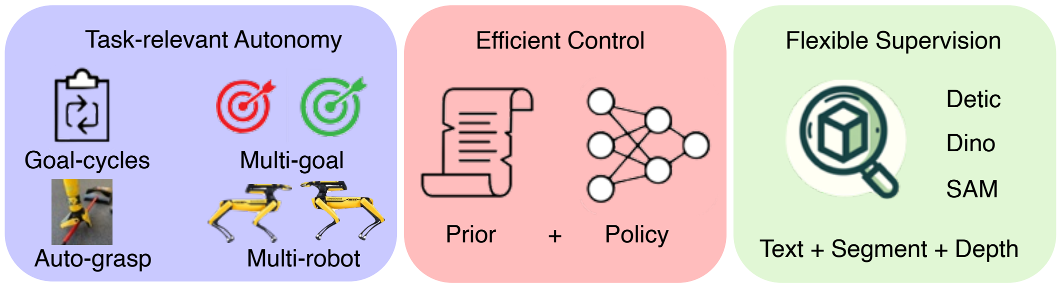


Figure 7.2: **Method Overview**: The main components of our approach for robots to continually practice tasks in the real world. **Left**: Task-relevant autonomy to ensure collection of useful data via object interaction, and maintaining state diversity via automated resets using multi-goal and multi-robot setups. **Center**: 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. **Right**: Flexible reward supervision that combines human-interpretable detection-segmentation with low-level, fine-grained depth observation.

7.3 Continuously Improving Mobile Manipulation via Real-world RL

Algorithm 6 Autonomous Mobile Manipulation

```
Require: Detection-segmentation models $M(.)$
Require: Behavior prior $P(.)$
 1: Initialize Data buffer $\mathcal{D}$, RL policy $\pi_\theta$
 2: Initialize task goal state $g_\mathcal{T}$
 3: Initialize trajectories per task $K$, horizon $H$
 4: while training do
 5:     for trajectory 1:K do
 6:         Deploy Auto-grasp/nav
 7:         for timestep 1:H do
 8:             Deploy policy $\pi_\theta(.)$ with prior $P(.)$
 9:             Compute reward $r_t$ using $M(o_t)$
10:             Add $(o_t, a_t, o_{t+1}, r_t) \mapsto \mathcal{D}$
11:             Sample batch $\beta \sim \mathcal{D}$
12:             Update $\pi$ with $\beta$ via RL
13:         end for
14:         (optional) If dist$(x, g_\mathcal{T}) \leq \epsilon$, break
15:     end for
16:     Switch task goal $g_\mathcal{T}$
17: end while
```

We design our approach to allow robots to autonomously practice and efficiently learn new skills without task demonstrations or simulation modeling, and with minimal human involvement. The overview of the approach we use is presented in Alg.6. Our approach has three components, as depicted in Fig 7.2: task-relevant autonomy, efficient control using behavior priors, and flexible reward specification. The first ensures the data collected is likely to have learning signal, the second utilizes signal from data to collect even better data to quickly improve the controller, and the third describes how to define learning signal for tasks. This allows learning difficult manipulation tasks, including tool use and constrained manipulation of large and heavy objects.

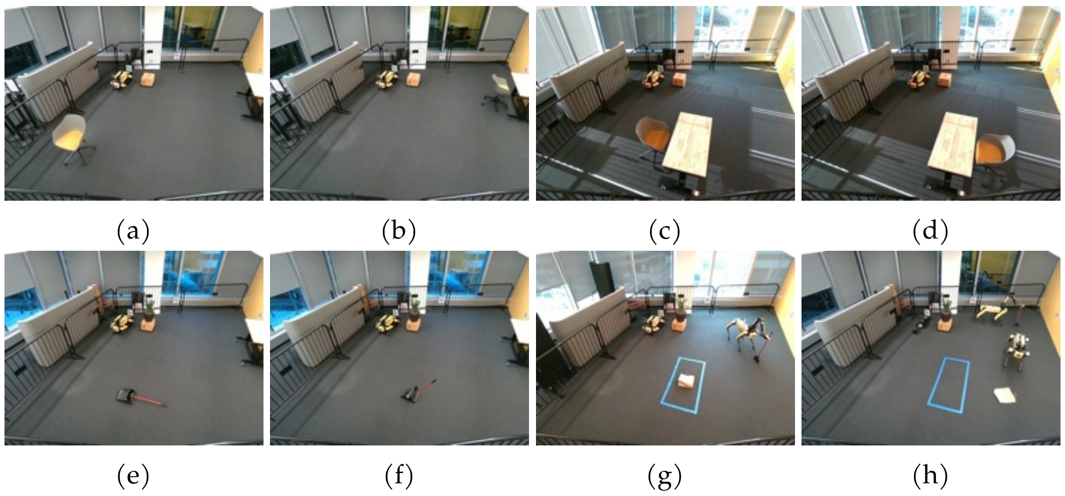


Figure 7.3: **Task Goals**: States that define goal-cycles for our 4 tasks, showing target position and orientation for the object of interest - (a-b): Chair Moving with a corner table, (c-d): Chair Moving with a middle table, (e-f): Long Handled Dustpan Standup, (g-h): Sweeping. Each setting requires the robot to alternate the object of interest (chair/dustpan/paper bag) between the pair of goal states.

7.3.1 Task-Relevant Autonomy

Auto-Grasp/Auto-Nav: For safe autonomous operation, we first create a map by walking the robot around the environment. This map is used by the robot to avoid collisions during its autonomous learning process. To ensure data collected involves object interaction, every episode begins with the robot estimating, moving to, and/or grasping the object of interest for the task. The object state is estimated using detection and segmentation models along with depth observations, as described in section 7.3.3. The robot then navigates towards the object position using RRT* to plan in SE(2) space using the collision map, and optionally deploys the grasping skill from the Boston Dynamics Spot SDK depending on the task. 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 grasp type, or on which portion of the object the grasp is performed. 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 prevent robot stagnation near goal states, we set up 'goal-cycles' within tasks, which serve as automated task resets. We show the different goal states used in each of the 4 tasks we consider in Fig.7.3. In the case of the chair moving tasks (Fig.7.3: a-d), the robot alternates between goals that are far apart in the x-y plane, and for the dustpan stand-up task (Fig.7.3 e,f), the robot needs to pickup the fallen dustpan and vertically orient and balance it. For the sweeping task (Fig.7.3: 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 previously described auto-grasp procedure.

7.3.2 Prior-guided Policy Learning

Incorporating Priors: We enable efficient learning by leveraging behavior 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 additional behavior needed to solve the task. 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 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 can deploy the prior and the policy in the following ways:
1. *Separate*: Trajectories are collected using either the prior $\{P(a_0|o_0), \ldots, P(a_T|o_T)\}$ or the policy $\{\pi(a_0|o_0), \ldots, \pi(a_T|o_T)\}$. 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. We make no assumptions on the optimality of the prior, and bootstrap learning via incorporating its *data*. In practice, we first collect trajectories using the prior, to initialize the data buffer for training the online RL policy $\pi(.)$.
2. *Sequential*: In addition to providing data with better signal 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:

$$\{P(a_0|o_0), .., P(a_L|o_L), \pi(a_{L+1}|o_{L+1}), .., \pi(a_T|o_T).\} \tag{7.1}$$

Thus, the prior structures the policy's initial state distribution, making learning easier. The data collected by the prior is 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. Such a prior can be used residually, where a policy adjusts the actions of the prior at every time step before being executed. These trajectories take the form:

$$\{P(a_0|o_0) + \pi(a_0|o_0), \ldots, P(a_T|o_T) + \pi(a_T|o_T)\} \tag{7.2}$$

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

$$J(\pi_\theta) = \mathbb{E}_{\substack{s_0 \sim p_0 \\ a_t \sim \pi_\theta(a_t|s_t) \\ s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)}} \left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t), \right] \tag{7.3}$$

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 [23]. RLPD is an off-policy method based on Soft-Actor Critic (SAC) [143], which samples from a mixture of data sources for online learning. Like REDQ [59], 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. Since our observations consist of raw images, we incorporate the image augmentations added by DrQ [440] to the base RL algorithm.

7.3.3 Flexible Supervision via Text-Prompted Segmentation

For flexible reward supervision, we combine semantic high-level information from vision and language models with low-level depth observations. Each task is defined by a desired configuration of some object of interest, so we derive a reward function by comparing the estimated state of the object at a given time to this desired state (see Section 7.4 for task-specific details). To estimate the state of the object, we start by using an open-vocabulary detection model Detic [462] to obtain the bounding box corresponding to the object of interest. We then obtain the corresponding object mask by conditioning a segmentation model, Segment-Anything [197], on the bounding box. Finally, using depth observations and the calibrated camera system for either the egocentric or fixed third-person cameras, we get a point cloud. Although this estimation is noisy, we find it sufficient to enable learning effective control policies via real-world RL. This system is flexible enough to handle different objects of interest, such as the chair, long handled dustpan for vertical orientation, or the paper bag for sweeping. Full details on the prompts, detection and segmentation models, and reward functions for each task in the supplemental materials.

7.4 Experimental Setup

For our experiments, we run continual autonomous RL using the Spot robot and arm system in a playpen of about 6×5 meters, enclosed with metal railings for safety. The playpen is mapped before autonomous operation to ensure the robot stays within bounds and doesn't collide with the railings. The navigation aspect of task autonomy involves searching for objects of interest. Since the main focus of this work is on learning complex manipulation skills, we do not use learning for the search problem; instead, we rely on a fixed camera in the scene. In addition to this, we also use the 5 egocentric body cameras of the Spot while searching for objects.

The chair-moving task requires the robot to grasp a chair and move it between goal locations. We consider two variants, chair-tablecorner (Fig.7.3 a-b) and chair-tablemiddle (Fig.7.3 c-d). The latter is more challenging since collisions between the chair and table base are much more frequent and the robot has to operate in a much tighter space. The dustpan standup task involves lifting up the long handle of a dust-pan (Fig.7.3-e), and then vertically balancing it so that it can stay upright on its base (Fig.7.3-f). Sweeping 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 (Fig.7.3-g). The other robot does not use learning, instead using the auto-grasp procedure to reset the paper bag by picking it up and dropping it close to the initial position(Fig.7.3-h). For each task, we specify success criteria for task completion, which corresponds to reaching the goal states in Fig.7.3. We list the choice of the prior, its combination with the policy, the state measurements used for reward, and reward sparsity in Table 7.1.

	Prior	Policy mode	Reward	Sparse
Chair-tablecorner	RRT*	Sequential	Chair-goal distance	False
Chair-tablemiddle	RRT*	Sequential	Chair-goal distance	False
Dustpan Standup	Scripted	Separate	Handle height	True
Sweeping	Distance constraint	Residual	Bag-goal distance	False

Table 7.1: We list the choice of prior, how it is combined with the policy, how reward relates to the object state, and whether the reward is sparse.

The observation space for RL policy training for all tasks consists of three 128X128 RGB image sources: the fixed, third-person camera and two egocentric cameras on the front of the robot. Additionally, we use the body position, hand position, and target goal. The action space for the chair and sweeping tasks is 5 dimensional, with base (x, y, θ) control and (x, y) control for the hand relative to the base. The dustpan stand-up task is 3 dimensional, consisting of $(z, \text{yaw}, \text{gripper})$ commands for the hand, where the gripper open action terminates the episode. We use the same network architectures for image processing, critic functions, policy, etc., for all comparisons. Please see supplementary materials for more details on the full reward functions, success criteria, procedural functions for priors, hyper-parameters for learning, and network details.

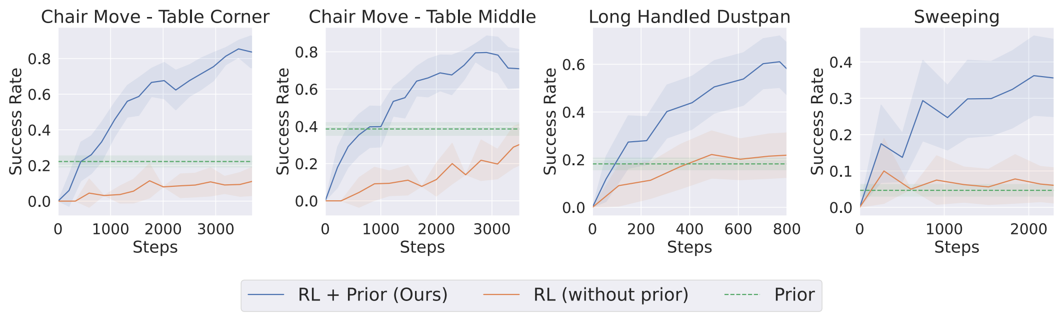


Figure 7.4: **Continual training improvement**: Success rate vs number of samples for ours, only RL and only prior. Note that we use our task-relevant autonomy approach with all methods. We see that our approach continuously improves with experience across tasks, learning much faster than RL without priors, and attaining significantly higher performance than just using the prior.

7.5 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 using only RL? 4) How does the policy learned via autonomous training perform when evaluated in test settings?

Task-relevant Autonomy: Running the robot without auto-grasp or goal-cycles, with the full action space comprising base and arm movement to any position in the playpen does not lead to any meaningful change in task progress even over long periods of time. Further, such operation is unsafe since the robot arm can get stuck in the enclosure railings, or strike the wall in an outstretched configuration. Hence, all the experiments we conduct, including those for baselines, utilize the task-relevant autonomy component so that the robot can make some progress on the task.

Continual Improvement via Practice: Given our task autonomy procedure, how effective is our proposed approach of combining real world RL with behavior priors, as opposed to using either only the prior or RL? From Fig.7.4, we see that our approach learns significantly faster than using only RL, and attains much superior performance than the prior, for each of the tasks. On the especially challenging sweeping task which involves tool use of the broom with a deformable paper bag, using only the prior or only RL leads to almost no progress, while our method is able to learn the task. Each robot training run takes around 8-15 hours, with the variation in time owing to different goal reset strategies across tasks and variance in how often the robot retries grasping objects for task-relevant autonomy. Hence, for fair comparisons across methods, we use the number of real-world samples collected to measure efficiency. The system also needs to be robust to many different factors in order to learn these tasks. The training area is exposed to sunlight, and the robot keeps collecting data and learning throughout the day with varying degrees of illumination. Object starting positions and grasps can vary widely, which affects the resulting object dynamics when practicing the task.

RL without Prior: For some tasks, using RL without the prior does improve in performance, but at a much slower rate than our method. Without the prior, RL often spends samples exploring parts of the state that are far from the goal. To illustrate this, we plot the average reward over each trajectory for the chair tasks (Fig.7.5). 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 our method executes the prior and policy sequentially for the chair task, our policy always starts out reasonably close to the goal, and can thus can pick up on high reward signal more often, leading to faster learning. We observe a similar pattern for the sweeping task, where using only RL leads the robot to wander around the playpen, greatly decreasing the likelihood of interacting with the paper bag and obtaining high reward.

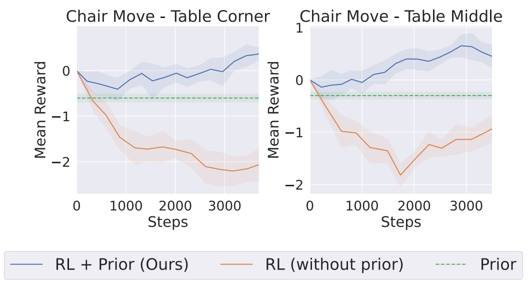


Figure 7.5: **Training mean reward**: Mean reward vs number of samples for the chair moving tasks. The negative average reward for RL without priors indicates that the robot is often far from the goal location.

Prior without RL: While the behavior priors are effective at bootstrapping learning, they are not sufficient on their own. This is because they do not adapt or learn from experience, and so keep repeating the same mistakes without improvement over time. We illustrate a qualitative failure example of the behavior prior for the chair moving task in Fig.7.6, where the robot following the RRT* planner runs into a collision state due to the simplified model being used. In contrast, our approach adapts the policy based on its experience to improve its performance, avoiding such collisions. For some tasks like sweeping the behavior prior is much simpler, only providing a constraint not to move too far away from the paper bag, which does not specify how the robot should sweep.

Final Policy Evaluation: We evaluate the final policies obtained after autonomous, continual practice and find that our approach obtains an average success rate of 80% across tasks from Table 7.2. For comparisons between our method and using only RL, we evaluate models obtained with the same number of real world samples. For evaluation, we use the deterministic policy instead of sampling from the stochastic distribution, which is used during training.

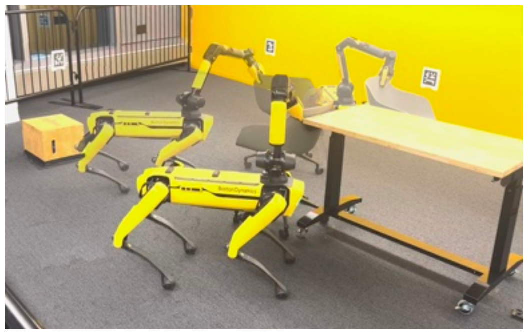
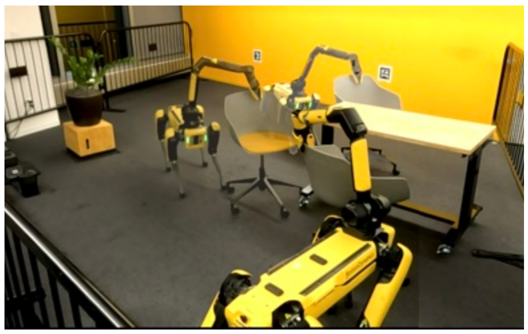



Figure 7.6: **Left**: The prior (RRT* with incomplete model) gets stuck in a collision with the table and is unable to recover as the planner does not have a model of chair-table interaction dynamics. **Right**: Our approach effectively recovers from collisions to complete the task.

	Ours	Only RL	Only Prior	Offline RL
Chair-tablecorner	**100%**	20%	22%	10%
Chair-tablemiddle	**80%**	50%	38%	20%
Dustpan Standup	**60%**	20%	18%	**60%**
Sweeping	**80%**	0%	5%	10%

Table 7.2: **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 4× improvement over using only the prior or only RL.

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.7.3-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.

Prior Data Quality: The behavior prior helps our approach in two ways, by structuring exploration for online learning, and also by providing higher quality data than random search, containing higher reward. To test the quality of the data obtained by the prior, we run offline RL on the dataset collected by the prior. This utilizes the reward of transitions to learn a policy, without any online rollouts. From Table 7.2, we see that on the chair and sweeping tasks, the behavior prior data quality is much worse, with an average success rate of 13%. The case of dustpan standup 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 learns strategies that are very different from the behavior prior, through exploration. This involves raising the robot's arm and dropping the dustpan, such that it lands upright. On the other hand, offline RL sticks close to the successful examples from the behavior prior generations.

7.6 Discussion and Limitations

We have presented an approach for continuously learning new mobile manipulation skills. This is enabled using task-relevant autonomy, efficient real-world control using behavior priors, and flexible reward definition. The current approach uses learning primarily for acquiring low-level manipulation skills after objects are grasped. Using automated procedures for navigation and search making use of a fixed third-person camera is a current limitation. This can be addressed by adding learning for the higher-level search problem too, which would allow the robot to rely just on its egocentric observations. This would allow learning in more unstructured, open-ended environments.

Chapter 8

Adaptive Mobile Manipulation for Articulated Objects in the Open World

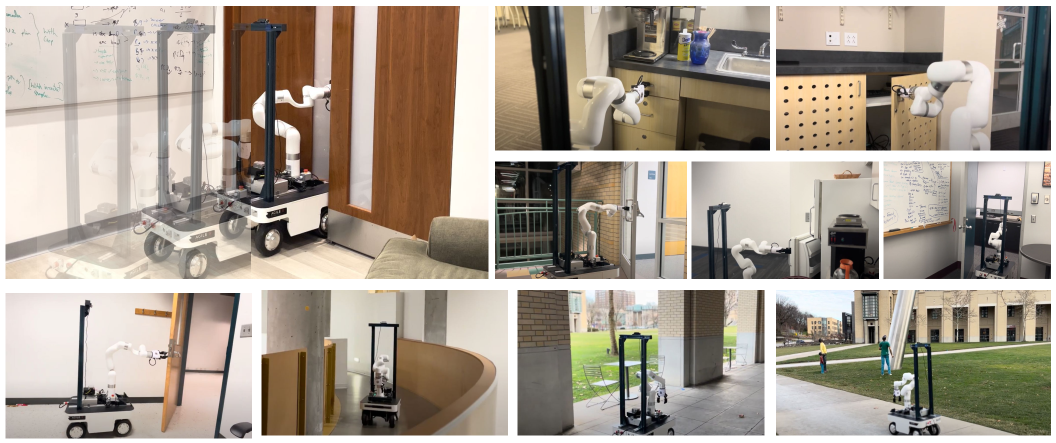


Figure 8.1: **Open-World Mobile Manipulation System**: We use a full-stack approach to handle articulated objects such as real-world doors, cabinets, drawers, and refrigerators in open-ended unstructured environments.

8.1 Motivation

Deploying robotic systems in unstructured environments, such as homes has been a long-standing research problem. Recently, significant progress has been made in learning-based systems [19, 206, 54, 354] towards this goal. However, this progress has been largely made independently either in mobility or in manipulation, while a wide range of practical robotic tasks require dealing with both aspects [116, 353, 37, 439]. The joint study of mobile manipulation paves the way for

generalist robots which can perform societally useful tasks [377] in open-ended unstructured environments, as opposed to being restricted to controlled laboratory settings focused primarily on tabletop manipulation. However, developing and deploying such robot systems in the *open-world* with the capability of handling unseen objects is challenging for a variety of reasons, ranging from the lack of capable mobile manipulator hardware systems to the difficulty of operating objects in diverse scenarios. Consequently, most of the recent mobile manipulation results end up being limited to pick-move-place tasks[444, 159, 383, 234], which is arguably representative of only a small fraction of problems in this space. We focus on operating everyday articulated objects, such as doors, drawers, refrigerators, or cabinets in open-world environments with a mobile manipulation robot. This is a common and essential task encountered in everyday life, and is a long-standing problem in the community [15, 85, 24, 65, 170, 269, 293].
The primary challenge is handling *unseen* objects across diverse varieties in unstructured open-world environments, rather than tabletop manipulation in a constrained lab setup. In this project, we take a *full-stack* approach to address this challenge. First, we provide a simple and intuitive solution to build a mobile manipulation hardware platform, enabling the transition from tabletop manipulation to mobile manipulation. The hardware follows two main principles - (1) versatility and agility which is essential to effectively operate diverse objects with different physical properties in potentially challenging environments, and (2) affordability and rapid-prototyping, since the system is assembled with off-the-shelf components, it is accessible and can readily be used by most research labs costing less than $25,000$ USD. We motivate our learning approach by considering how people typically approach operating articulated objects like doors: this process generally starts with reaching towards a part of the object, such as a handle, and establishing a grasp, then follows by constrained manipulation primitives such as rotating, unlocking, or unhooking, where arm or body movements are applied to operate the object. In addition to this high-level strategy, there are also lower-level decisions made at each step regarding the exact direction of movement, extent of perturbation, and amount of force applied. Inspired by this, we introduce *primitives as APIs* for efficient learning in our framework, which includes a grasp primitive API utilizing pretrained visual models, and a constrained mobile-manipulation primitive API. These primitives are parameterized by *learned* parameters, which need to be adapted online to operate diverse articulated objects. We start by initializing a single policy using a set of expert demonstrations through behavior cloning (BC). The policy takes visual inputs and outputs the parameters of the primitive APIs, as shown in the Figure 8.6. The central challenge we face is operating with objects that fall outside the BC training domain. For instance, with visually ambiguous objects, it can be difficult to determine whether a door needs to be 'pulled' or 'pushed' based solely on visual observation. Furthermore, if all the doors in the expert demonstrations are 'pull' doors, the behavior cloning policy may struggle to generalize to 'push' doors that it has not previously encountered.

To address this, we develop a system capable of fully autonomous online adaptation using reinforcement learning (RL), allowing the robot to keep on improving its performance without any human intervention. In our online adaptation, we use safety aware exploration, where we monitor the robot arm joint current to avoid unsafe actions, and leverage this to create negative rewards. Learning requires real-world reward specification, using vision language models (VLMs). For autonomous practice we use a self-reset mechanism, where we employ visual odometry to enable the robot to navigate back to its initial position.
In this Chapter, we present Open-World Mobile Manipulation System, a full-stack approach to tackle a challenging problem: mobile manipulation of diverse articulated objects in unstructured open world environments. To validate the effectiveness and practicality of our system, we conducted a field test of 8 novel testing objects ranging across 4 buildings on a university campus to test the effectiveness of our system, and found adaptive learning boosts system performance after the online adaptation.

8.2 Related Work

Adaptive Real-world Robot Learning: Prior work has demonstrated how robots acquire novel behavior by using real-world reinforcement learning with rewards [220, 221, 182, 183], and even with unsupervised exploration [295, 251, 20]. Another line of research has been dedicated to utilizing real-world online data to infer environmental parameters or latent representations for adapting policies [315, 322, 225, 62, 263]. More recently, there have been approaches that use RL to fine-tune initialized offline learning policies [149, 148, 435, 290]. Other methods aim to do so without access to demonstrations on the test objects, and pretrain using other sources of data - either using offline robot datasets [209], simulation [371] or human video [252, 184, 431, 430] or a combination of these approaches [159]. We operate in a similar setting, without any demonstrations on test objects, and focus on demonstrating RL adaptation on mobile manipulation systems that can be deployed in open-world environments, bootstrapping from a set of demonstrations on a training set.

Learning-based Mobile Manipulation Systems: In recent years, the setup for mobile manipulation tasks in both simulated and real-world environments has been a prominent topic of research [422, 445, 377, 338, 420, 257, 460, 115, 116, 37, 165]. Notably, several studies have explored the potential of integrating Large Language Models into personalized home robots [424, 5, 37]. While these systems display impressive long-horizon capabilities using language for planning, these assume fixed low-level primitives for control. In our work, we seek to learn low-level control parameters via interaction. Furthermore, unlike the majority of prior research which predominantly focuses on pick-move-place tasks [444], we consider operating articulated objects in unstructured environments.

Door Manipulation: Research in door opening has a rich history in the robotics community [65, 170, 269, 293]. A significant milestone was the DARPA Robotics Challenge (DRC) finals in 2015 [15, 85, 24]. Nevertheless, prior to the deep learning era, the primary impediment was the robots' perception capabilities [333]. Recent approaches using deep learning to address vision challenges include Wang et al. [410], which leverages synthetic data to train keypoint representation for the grasping pose estimation and Qin et al. [303], which proposed an end-end point cloud RL framework for sim2real transfer. The prospect of large-scale RL combined with sim-to-real transfer holds great promise for generalizing to a diverse range of doors in real-world settings [399, 138, 303]. However, such policies might struggle when faced with an unseen door with physical properties, texture, or shape different from the training distribution. Our approach can keep on learning via real-world online samples, and can learn to adapt to difficulties faced in operating new unseen doors.

8.3 Open-world Mobile Manipulation Systems

In this section, we detail our *full-stack* approach, which includes hardware integration, and the adaptive learning framework. This enables our mobile manipulation system to learn adaptively in open-world environments, allowing it to operate everyday articulated objects such as cabinets, drawers, refrigerators, and doors. More details and all the hyperparameters we used are provided in the supplementary materials.

8.3.1 Hardware

The transition from tabletop manipulation to mobile manipulation is challenging not only from algorithmic studies but also from the perspective of hardware. In this project, we provide a simple and intuitive solution to build a mobile manipulation hardware platform. Everyday articulated objects like doors exhibit a wide range of physical properties, including weight, friction, and resistance. To handle these effectively, we specifically look for a platform with high payload capabilities to interact with heavy doors, which requires versatility and agility. Additionally, we aimed to develop a human-sized platform with an omnidirectional base capable of maneuvering across various real-world doors in unstructured and narrow environments. The platform is designed to be low-cost and employs off-the-shelf components, which enables rapid prototyping.

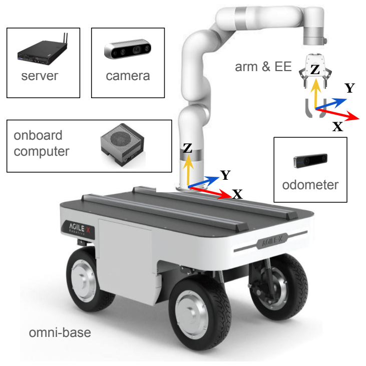


Figure 8.2: **Mobile Manipulation Platform:** A cost-effective and user-friendly hardware platform for research.

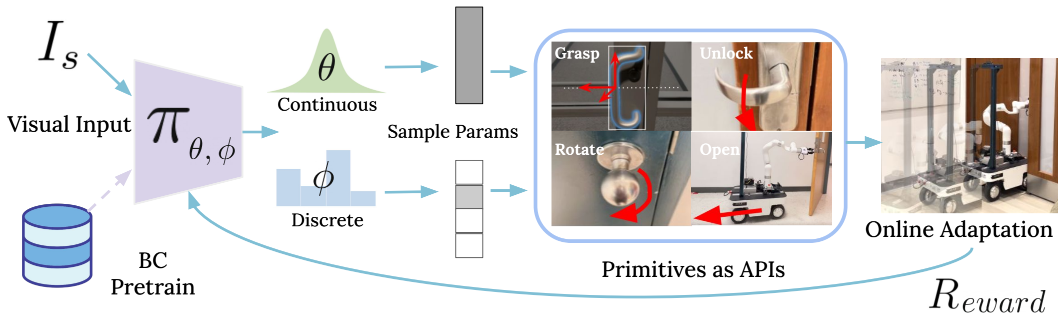


Figure 8.4: **Adaptive Learning Framework:** We begin by initializing a single policy using a set of expert demonstrations through behavior cloning. The policy takes visual inputs and outputs continuous and discrete parameters sampled by Gaussian and categorical distributions. To enable efficient learning, primitive APIs are utilized. They take the sampled parameters as input and instantiate action executions. To address new objects that fall outside the training domain, we develop a system capable of fully autonomous Reinforcement Learning (RL) adaptation to keep on improving the system's performance.

8.3.2 Task Definition

In this project, we consider a set of articulated objects that consist of three rigid parts: a base part, a frame part, and a handle part. The base and frame are connected by either a revolute joint (as in a cabinet) or a prismatic joint (as in a drawer). The frame is connected to the handle by either a revolute joint or a fixed joint. This covers objects such as doors, cabinets, drawers, and fridges. As shown in Figure 8.3, we identify four major types of the articulated objects: Handle articulations commonly include levers (Type A) and knobs (Type B). For cases where handles are not articulated, the frame part can either revolve around a hinge using a revolute joint (Type C), or slide back and forth along a prismatic joint, for example, drawers (Type D). This categorization covers a wide variety of everyday articulated objects.

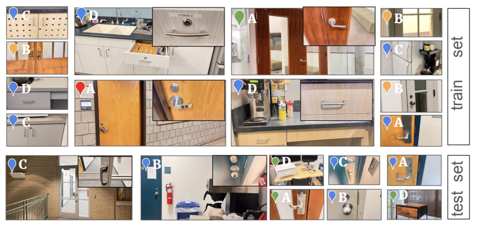


Figure 8.3: **Articulated Objects**: Visualization of the 12 training and 8 testing objects we used, with type labeled, and with indicators corresponding to locations.

8.3.3 Adaptive Learning Framework

We start by initializing a single policy using a limited set of expert demonstrations of the four types of articulated objects through behavior cloning. The central challenge we face is operating with new objects that fall outside the BC training domain. To address this, we develop a system capable of fully autonomous online adaptation using reinforcement learning (RL), allowing the robot to *keep on improving* its performance without any human intervention.

8.3.3.1 Primitives as APIs

Inspired by the prior works of efficient policy learning with manipulation primitive priors [274, 141], we prebuilt a set of expressive primitives. Each primitive is a functional API that takes *learned* parameters as the input to instantiate an action execution. A trajectory of fixed horizon is executed in an open-loop manner: a grasping primitive followed by a sequence of N constrained mobile-manipulation primitives, denoted as: $\{I_s, G(g), \{M(C_i, c_i)\}_{i=1}^{N}, I_f, R\}$, where I_s is the initial observed image, $G(g)$, $M(C_i, c_i))$ denote the parameterized grasp and constrained manipulation primitives respectively, I_f is the final observed image, and r is the reward for the trajectory.

Grasp Primitive with Passive Visual Models: We structure a vanilla 6D grasp pose by using *passive* visual models of detection and segmentation [462, 197], as shown in the ”Primitives as APIs” block of Figure 8.6. Given a text prompt of ”handle”, the open-vocabulary detection model reliably returns a 2D bounding box of the handle. We use a heuristic to formulate the orientation of the grasp pose: If the width of the 2D bounding box is smaller than the length of the bounding box, we determine it is a vertical handle; otherwise, it is a horizontal handle. The grasp orientation is further determined using the surface normal estimation of the door frame, the vertical-horizontal type of the handle, and the direction of gravity. For XYZ position of the grasp pose: we draw a middle line to locate the center point of the segmentation mask of the handle. We then compute the x,y,z potion in camera frame with depth. The camera frame is calibrated with robot base frame, so we can transform the x,y,z position to the robot base frame. However, passive detection and segmentation models are insufficient to predict a robust grasp pose for all handle types. For our grasp primitive, we introduce a 3-dimension continuous low-level parameter ranging from -1 to 1 as the grasp primitive input, which is then rescaled to the grasp residuals ranging from $-d$ to d. This is beneficial since our grasp residuals can be adjusted to diverse handles via online adaptation. We denote the grasping primitive API as $G(.)$, which is parameterized by the parameter g.

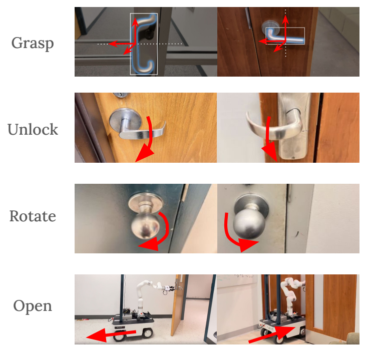


Figure 8.5: **Primitives**: We show visualization of the primitives we used in our adaptive learning framework.

Constrained Mobile-Manipulation Primitive: As shown in Figure.8.2, we define two coordinate frames in the mobile manipulation system, a base frame, and an arm end-effector frame. The arm end-effector frame is defined relative to (i.e. with respect to) the base frame. With a 3-DOF motion for the base (in the SE(2) plane), and a 6-DOF arm (with respect to the base frame), we have a 9-dimensional vector – $(v_x, v_y, v_z, v_{\text{yaw}}, v_{\text{pitch}}, v_{\text{roll}}, V_{\text{x}}, V_{\text{y}}, V_\omega)$. The first 6 dimensions correspond to velocity control for the arm end-effector, and the last three are the velocity control for the base. The primitives we use impose contraints on this space as follows - Unlock : $(0, 0, v_z, v_{\text{yaw}}, 0, 0, 0, 0, 0)$; Rotate : $(0, 0, 0, v_{\text{yaw}}, 0, 0, 0, 0, 0)$; Open : $(0, 0, 0, 0, 0, 0, V_{\text{x}}, 0, 0)$; as shown in the "Primitives as APIs" block of Figure 8.6. The constrained mobile-manipulation API takes two parameters, a discrete parameter C to select one of these primitives to take, and a continuous parameter c to determine the low-level control of the selected primitive. More details are provided in the supplementary materials.

8.3.3.2 BC Pretraining

Algorithm 7 Adaptive Learning

```
Require: Grasping primitive $G(.)$ taking param $g$
Require: Constrained manipulation primitives $M(.)$, taking parameter C and c.
 1: Initialize primitive classifier $\pi_\phi(\{C_i\}_{i=1}^N|I)$
 2: Initialize conditional policy network $\pi_\theta(g, \{c_i\}_{i=1}^N|I, \{C_i\}_{i=1}^N)$
 3: Collect a dataset $D$ of expert demos $\{I, g, \{C_i\}_{i=1}^N, \{c_i\}_{i=1}^N\}$
 4: Train $\pi_\phi$ and $\pi_\theta$ on $D$ using Imitation Learning
 5: for online RL iteration 1:$N_{iter}$ do
 6:     for sampling rollout 1:$N_{rol}$ do
 7:         Obtain observation image $I_s$
 8:         Sample $\{C_i\}_{i=1}^N \sim \pi_\phi(.|I_s)$
 9:         Sample $(g, \{c_i\}_{i=1}^N) \sim \pi_\theta(.|I_s)$
10:         Execute traj $\{G(g), \{M(C_i, c_i)\}_{i=1}^N\}$
11:     end for
12:     Update policies $\pi_\phi$ and $\pi_\theta$ using RL
13: end for
```

Imitation learning has been proven effective in a variety of robotics applications [431, 430, 116, 19]. In this project, we start by initializing our policy using a limited set of expert demonstrations through behavior cloning. Given an initial observation image I_s, we use a primitive classifier $\pi_\phi(\{C_i\}_{i=1}^N|I)$ to predict a sequence of N discrete parameters $\{C_i\}_{i=1}^N$ for constrained mobile-manipulation, and a conditional policy network $\pi_\theta(g, \{c_i\}_{i=1}^N|I, \{C_i\}_{i=1}^N)$ which produces the continuous parameters of the grasping primitive and the constrained mobile-manipulation primitives. We start by initializing our policy using a small set of expert demonstrations via behavior cloning. We collect an offline demonstration dataset by teleoperating the mobile manipulation robot in the open world. We type the keyboard to select the primitives and long-press the keyboard bottom to instance the low-level parameters. The imitation learning objective is to learn $\pi_{\theta,\phi}$ that maximize the likelihood of the expert actions. Specifically, given a dataset of image observations I_s, and corresponding actions $\{g, \{C_i\}_{i=1}^N, \{c_i\}_{i=1}^N\}$, the imitation learning objective is: $\max_{\phi,\theta} \left[\log \pi_\phi(\{C_i\}_{i=1}^N \mid I_s) + \log \pi_\theta(g, \{c_i\}_{i=1}^N \mid \{C_i\}_{i=1}^N, I_s)\right]$. We include 3 objects from the 4 category in the BC pretraining dataset, collecting 10 demonstrations for each object, producing a total of 120 trajectories. We pretrain a single BC policy with all the 120 training objects. We also have 2 held-out unseen objects from each category as testing objects for experiments. The training and testing objects differ significantly in visual appearance (eg. texture, color), physical dynamics (eg. if spring-loaded), and actuation (e.g. the handle joint might be clockwise or counter-clockwise). We include visualizations of all objects used in train and test sets in Fig. 8.3.

8.3.3.3 Online RL Adaptation

Generalizing to unseen objects using a vanilla behavior cloning (BC) model is extremely challenging. To address such out-of-distribution scenarios, we enable the policy to keep on improving via online adaptation. Specifically, for each testing object, we fine-tune the pre-trained BC policy with online sampled data. This corresponds to maximizing the expected sum of rewards under the policy: $\max_{\theta,\phi} \mathbb{E}_{\pi_{\theta,\phi}} \left[\sum_{t=0}^{T} r(s_t, a_t)\right]$. Since we utilize a highly structured action space with primitives as described previously, we can optimize this objective using a fairly simple RL algorithm. Specifically we use the REINFORCE objective [419]: $\nabla_{\theta,\phi} J(\theta, \phi) = \mathbb{E}_{\pi_{\theta,\phi}} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi(a_t|s_t) \cdot r_t\right] = \mathbb{E}_{\pi_{\phi,\theta}} \left[(\nabla_\phi \log \pi_\phi(C_i|I) + \nabla_\theta \log \pi_\theta(g, c_i|C_i, I)) \cdot R\right]$, where R is the reward provided at the end of trajectory execution. Note that we only have a single time-step transition, all actions are determined from the observed image I_s, and executed in an open-loop manner. We use a weighted objective to avoid overfitting during fine-tuning, where the overall loss is $\mathcal{L}_{\text{overall}} = \mathcal{L}_{\text{online}} + \alpha * \mathcal{L}_{\text{offline}}$. Note that we train separate RL policies for each testing object; we leave the lifelong learning scheme for future work. The learning procedure is concluded in the algorithm 7.

Safety Aware Exploration: Deploying our system with online sampling necessitates the avoidance of dangerous actions. We use a safety mechanism based on monitoring the robot arm's joint current during online sampling. If the robot samples an action that causes the joint current to meet its threshold, we terminate the episode and reset the robot to prevent potential self-damage. Additionally, we leverage this to provide negative rewards to disincentivize such actions.

Real-World Reward Specification: A naive approach to reward specification involves a human operator providing a binary reward, where the reward is +1 if the robot successfully opens the doors, and 0 if it fails or if a safety violation occurs. Manual reward annotation is feasible since the system requires very few samples for learning. However, to enable a fully autonomous learning system, we aim to remove the bottleneck of relying on human presence in the loop. We investigate leveraging large vision language models (VLMs) as a source of automatic reward. Specifically, we utilize CLIP [308] to compute the probability of the encoded visual features of an image input matching the encoded text features of two text prompts: *"door that is closed"* and *"door that is open"*. We then record the probability of *"door that is open"* for the input image and denote it as the door opening score. We capture two images from the same camera viewpoint: one at the start of the online sampling rollout and another at the end. If the door opening score of the last image is higher than that of the start image (indicating an actually open door, which always holds true in our experimental findings), we assign the robot a positive reward of 1. If a safety protection is triggered, the reward is -1; otherwise, it is 0.

Self-Reset Mechanism: The robot employs visual odometry, utilizing a realsense T265 tracking camera mounted on its base, enabling it to navigate back to its initial position. At the end of every episode, the robot releases its gripper, and moves back to the original SE(2) base position, and takes an image of I_f for computing reward. We then apply a random perturbation to the SE(2) position of the base. Furthermore, if the reward is 0, where the door is still closed, the robot goes to the next episode. If the reward is 1, where the door is open, the robot base uses a script policy where the robot base moves forward and gently closes the door by making contact with it.

8.3.4 Model Parameterization

In our setup, the policy includes a primitive classifier π_ϕ, and a conditional policy network π_θ. They share a frozen visual backbone, which is a ResNet-18 [157] pre-trained on ImageNet [90]. We begin by cropping the handle from the RGB image and padding the cropped image into a square shape with zero-padding. The visual backbone takes this processed RGB image as the input. The primitive classifier outputs discrete actions sampled from categorical distributions, determining a sequence of primitive types, and the conditional policy network takes both visual inputs and the discrete actions from the primitive classifier, and outputs continuous parameters, sampled from gaussian distributions for the respective primitives, together instantiating a sequence of primitives. To ensure that the action sequence produced by the open-loop policy maintains a fixed horizon, we have introduced a blank primitive during policy learning, which allows for the skipping of action execution.

8.4 Experimental Results

We conduct an extensive field study involving 12 training objects and 8 testing objects across four distinct buildings on the university campus to test the efficacy of our system. In our experiments, we seek to answer the following questions: 1) Can the system improve performance on unseen objects via online adaptation across diverse object categories? 2) How does this compare to simply using imitation learning on provided demonstrations 3) Can we automate providing rewards using off-the-shelf vision-language models?

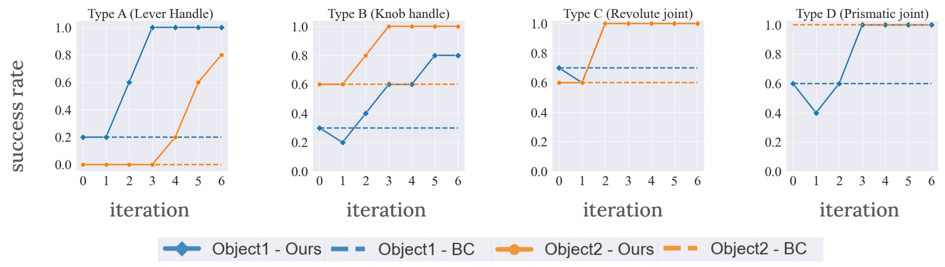


Figure 8.6: **Online Improvement**: Our adaptive approach is able to improve in performance, while the imitation policy has limited generalization. For the iteration axis, 0 means the pretrained BC model evaluation success rate. We conduct online sampling and model updating from iteration 1 to 5, and report the success rate of the online sample data in the graph. The last success rate data point of iteration 6 in the graph is the evaluation success rate of the final fine-tuned model.

8.4.1 Online Improvement

We evaluate our approach on 4 categories of held-out articulated objects, two test objects from each category. We define a successful articulated object opening rollout as one where a sufficient gap is created between the frame part and the base part to ensure that: 1) Human experts can visually confirm the presence of a gap. 2) For door opening, the robot base can traverse through the door under human teleoperation.

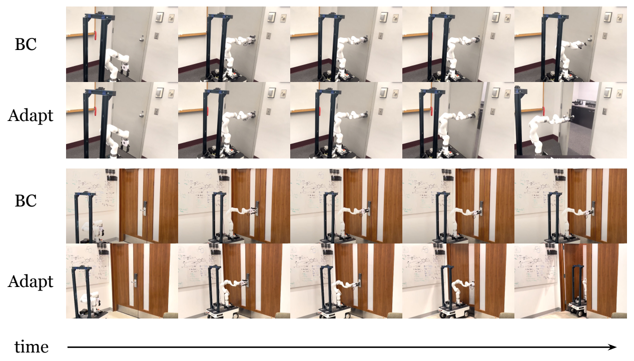


Figure 8.7: **Online Improvement Visualization**: In the third row, the pretrained BC policy completely fails to open the ”push” door. With continual practice, the robot is able to achieve a high success rate after online adaptation in the fourth row. For doors with knobs, in the first row, using the vanilla grasp does not establish a robust grasp pose, our adaptation method enables the robot to adjust the grasp pose through online adaptation.

At the testing time, the robot is initialized randomly in front of the objects. Given the RGBD image of the scene obtained from the realsense camera, we use off-the-shelf visual models [462, 197] to obtain the mask of the door frame using just text prompts. Furthermore, since the door is a flat plane, we can estimate the surface normals of the door using the corresponding mask and the depth image. This is used to move the base close to the door and align it perpendicularly, to start the policy rollout.

We first evaluate the pretrained BC model on the unseen testing objects. We report the the success rate of 5 rollouts as shown in the dotted line in Fig. 8.6. Then we go to the online adaptation stage, where we fine-tune the pretrained BC model with the online samples collected on the testing objects using human labeled reward, which is a score from -1 to 1. We report continual adaptation performance in Fig. 8.6. From Fig. 8.6, we see that our approach improves the average success rate across all objects from 50 to 95 percent. Hence, continually learning via online interaction samples is able to overcome the limited generalization ability of the initial behavior cloned policy. In cases where the BC policy is reasonably performant, such as Type C and D objects with an average success rate of around 70 percent, RL is able to perfect the policy to 100 percent performance. Furthermore, RL is also able to learn how to operate objects even when the initial policy is mostly unable to perform the task. This can be seen from the Type A experiments, as visualized in Fig. 8.7, where the pretrained BC policy completely fails to open the ”push” door (Object2). That is because all the BC training objects do not require ”push” $(V_x > 0)$ in the open primitive. With continual practice, RL is able to achieve a success rate of 80 percent. This shows that RL can explore actions that are potentially out of distribution from the BC pretraining dataset, and learn from them, allowing the robot to learn how to operate novel unseen articulated objects. We also observed that for doors with knobs, as shown in Figure 8.7, using detection and segmentation models does not provide the robot with an ideal grasping pose. Our adaptation method enables the robot to adjust the grasp pose through online interactions, significantly increasing the success rate from 45% to 90%.

8.4.2 Action-replay baseline

We introduce a baseline that involves replaying trajectories from the closest object in the training set. This closest object can be found using k-nearest neighbors with some distance metric [279]. This approach is likely to perform well especially if the distribution gap between training and test objects is small, allowing the same actions to be effective. We run this baseline for two objects that are particularly hard for behavior cloning, one each from Type A and B categories (lever and knob handles respectively). The distance metric we use to find the nearest neighbor in the training set is Euclidean distance of the the CLIP encoding of observed images. We evaluate this baseline both in an open-loop and closed-loop manner. In the former case, only the first observed image is used for comparison and the entire retrieved action sequence is executed, and in the latter we search for the closest neighbor after every step of execution and perform the corresponding action. From Table 8.8 we see that this approach is quite ineffective, further underscoring the distribution gap between the training and test objects in our experiments.

Action-Replay Comparison

	KNN-open	KNN-close	BC-0	Adapt-GT
Success Rate B1 (knob)	10%	0%	30%	80%
Success Rate A2 (lever)	0%	0%	0%	80%

Figure 8.8: Comparisons of the adaptation policies and BC policies with KNN baselines.

8.4.3 Autonomous reward via VLMs

We investigate whether we can replace the human operator with an automated procedure to provide rewards with VLMs to enable fully autonomous adaptation. We see that online adaptation with VLM reward achieves a similar performance as using ground-truth human-labeled reward, with an average of 80 percent compared to 90 percent. We report the performance after every iteration of online adaptation in Fig. 8.9. Removing the need for a human operator to be present in the learning loop opens up the possibility for autonomous training and improvement.

We also observed a drop in performance over iteration $1, 2, 3$ in Fig. 8.9. This can be attributed to the high variance of reinforce algorithm, which typically leads to unstable training in performance, further exacerbated by a small sample size.

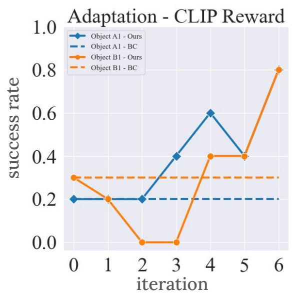


Figure 8.9: Online Adaptation using CLIP rewards, showing fully autonomous adaptation.

8.5 Limitations and Conclusion

The success of a door opening task relies on the initial accuracy of handle detection. However, when multiple handles are present in a scene, the detection model fails to accurately identify the specific handle of interest. Real-world door-opening tasks are challenging, as we discovered in our experiments where hardware proved to be one of the bottlenecks. 5 kg end-effector payload is a minimum requirement for common doors in everyday life. We present a full-stack system for operating various articulated objects in the open world. Our approach is able to keep on improving its performance across diverse articulated objects from different buildings across the university campus. The system can also learn using rewards from VLMs without human intervention, allowing for fully autonomous adaptation.

Part IV

Scaling Data for Robot Learning

Chapter 9

Bimanual Dexterity for Complex Tasks

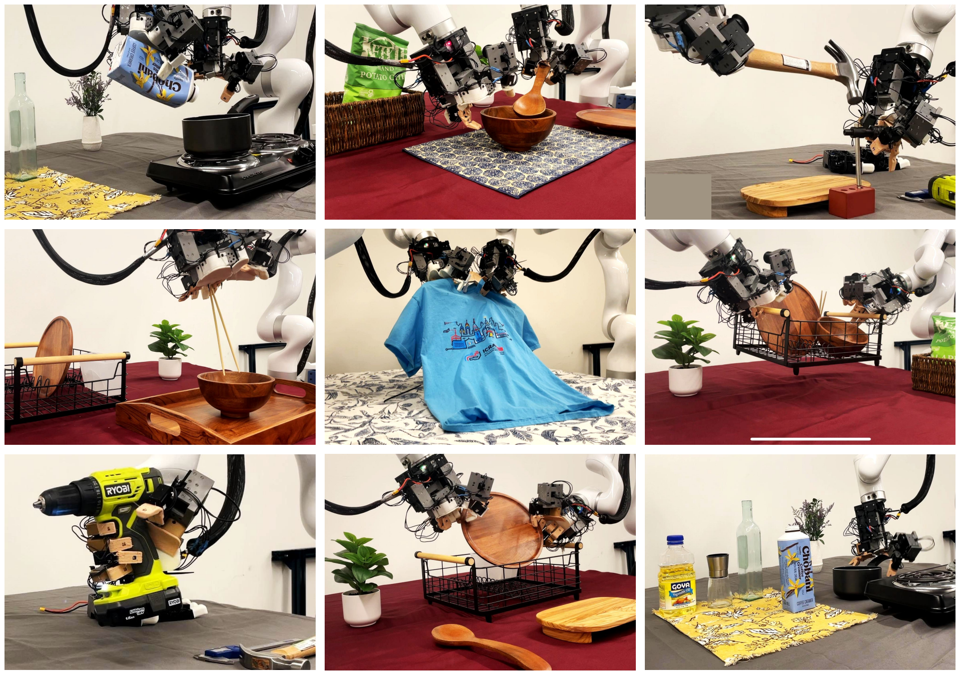


Figure 9.1: **Bimanual Dexterity:** Our teleoperation data collection system can perform various complex tasks including pouring, hammering, chopstick picking, hanger picking, picking up basket, drilling, plate pickup and pot picking, using a bimanual dexterous system.

9.1 Motivation

General-purpose robots that are deployed to perform everyday tasks will need to perform a wide variety of challenging manipulation tasks. This ranges from intricate movements like screwing in small objects, to cutting vegetables, to being able to move large objects like furniture. These are tasks built around humans, and correspond to activities that people can perform. The versatility of human hands in particular is essential for finer-grained tasks ranging from writing and creating art to typing on keyboards. Hence one approach to building such systems is to use a hardware form factor that resembles humans with two arms each equipped with a dexterous multi-fingered hand. With the advent of data-driven machine learning methods and low-cost hardware there has been renewed interest in humanoids and dexterous hands. There is great promise for machine learning approaches to enable effective control for high-dimensional robot systems using large amounts of data [221, 6, 168, 206, 182]. A key question remains: how do we go about collecting this high-quality data for bimanual robots? The key characteristics for an effective data collection system include being low-cost, easy to setup and use, low-latency and most importantly accurate enough, such that the data collected can be used to train robots to perform complex tasks of interest.

To address this problem, VR headsets have become increasingly prevalent due to their easy-to-use internal closed-source body tracking systems [92, 282]. However we find that wrist tracking is often jittery and the finger tracking is inaccurate. To mitigate this, SteamVR [401] uses LiDAR which provides less noisy estimates, but requires external tracking devices which doesn't allow for data collection for mobile robot setups. For even higher fidelity readings, there is work that uses motion capture and reflective marker-based approaches such as Vicon or Optitrack[403] but they are extremely expensive and difficult to setup, and suffer from the same fixed-location problem as SteamVR since they require many external cameras. An aspect of motion capture technology that has been used in robot learning in recent years are wearable gloves [408, 362, 246] for hand tracking which records the fingertip position of the human hand using EMF sensors. We use the Manus glove as part of our system, which is a professional Mocap glove with great accuracy and reliability.

For arm tracking, researchers in the robotics community have been recently using joint-level teleoperation for 2-finger grippers. [458, 426]. Wu et al. [426], find that a low-cost 3D printed scaled teacher arm model that has the same kinematic link structure of a large robot arm can be used for accurate and effective teleoperation. These methods only provide one DOF of finger tracking instead of the twenty two plus DOF of the human hand. Our key insight is to develop a system that combines this joint-based arm tracking along with a Mocap fingertip glove to achieve accurate low-cost teleoperation of an arm and hand system.

Our main contribution is BiDex, a system for dexterous low-cost teleoperation system for bimanual hands and arms in-the-wild in any environment. To operate, the user wears two motion capture gloves and moves naturally to complete everyday dexterous tasks. The gloves captures accurate finger tracking to map motions naturally to the robot hands and the GELLO [426] inspired arm tracking accurately tracks the human wrist position and joint angles for the robot arm. The data collected by the system can be used to train effective policies using imitation learning, after which the bimanual robot system can perform the task autonomously. BiDex costs around $5k for a pair of Manus gloves and few hundred dollars for the arm teleoperation which works for many existing robot arms. Even including the robot arms and robot hands used in our demonstration, the total cost is around $30k. We compare the accuracy and speed of data collection to other systems: the VR headset and SteamVR.

9.2 Related Work

Robot Arm Teleop: Many common approaches to teleoperation include using joysticks, space mouses, vision-based methods, [320], and VR headsets [13, 282, 92] which control arms with inverse kinematics. Joint based teleoperated control has been used in areas such as kinesthetic teaching, Brantner and Khatib [35], ABB YuMi [224] and Da Vinci Machines [112], and recently [458, 116] introduced a low-cost version of this with mirroring Trossen Robot arms. GELLO [426] uses light and inexpensive teacher arms that are 3D printed to control full size robot arms, a system we use in our BiDex due to its lost-cost, accurate, portable design.
Robot Hand Teleop: The high dimensionality makes tracking human hands particularly difficult. To control robot hands, many vision-based techniques such as [151, 304, 369] do not require specialized equipment but are not that accurate. Shadow Hand developed a professional system that uses SteamVR and two gloves to control two Shadow Hands [352]. Recently, bimanual robot hands and tracking have become accessible for academic labs. Dexcap uses LEAP Hand and tracks the human with gloves and a SLAM-based robot camera. [408] Hato uses a VR headset and controller to control two 6 DOF Psyonic Hands [227, 300]. A key question in controlling robot hands is how to map the human hand configuration to robot hand joints. These papers introduce inverse kinematics based methods that optimize pinch grasps between the human and robot hands [408, 151, 369, 304].
Motion Capture: Motion capture and graphics contributions often are useful in the robotic teleoperation domain. Outside-in mocap approaches use external sensing technology to track the human body or other objects in the scene. SteamVR uses external lasers and worn wireless laser receivers. [401] Vicon-based systems use reflective balls and external cameras to track. Inside-out approaches such as XSens [260] or Rokoko suit rely on IMUs on the body but these often drift over time and require recalibration [326, 227]. For hand data, many vision-based approaches such as Frankmocap [329] return MANO [328] parameters which can be converted to robot hand joint angles.
Learning from Expert Demonstrations: Recently the robot learning community has seen notable success in learning from demonstrations driven by the development of imitation learning algorithms [245, 63]. Complementing these advances, significant efforts have been made to scale up robotic datasets to facilitate more capable robotic systems [82, 278, 192, 406]. Despite these efforts, acquiring robotics data remains an expensive and challenging endeavor. To address these issues, developments

in low-cost hardware have been instrumental in democratizing access to robotic technology, enabling more widespread research and application [375, 458, 116, 64]. However, these systems are primarily focused on simple gripper functionalities; and the challenge of achieving more intricate dexterity and intuitive control in robotic systems motivates our bimanual dexterous teleop system.

9.3 Bimanual Robot Hand and Arm System

We introduce BiDex, a system designed to enable an operator to naturally teleoperate any bimanual robot hand and arm setup. BiDex is exceptionally precise, low-cost, low-latency, portable and can control any human-like pair of dexterous hands even with over 20 degrees of freedom. It achieves this accurate tracking of the human hand by utilizing a Manus VR glove based-system [247] and the human arm using a GELLO-inspired system [426]. We present the full process by which we send commands using our system for controlling bimanual hands with dexterous hands in Alg.8. Crucially, our solution is tailored to operate seamlessly in both tabletop and mobile environments because it does not need any external tracking devices and is very portable. In Section 10.5 we find that our system is highly intuitive, precise, and cost-effective compared to many commonly used approaches today such as the VR headset and SteamVR for both the tabletop and mobile manipulation settings on two different pairs of robot hands.

9.3.1 Multi-fingered Hand Tracking

A hand tracking approach must return skeleton information about the human hand which has over 20 degrees of freedom with low-latency. Many vision-based tracking systems today based on FrankMocap[329, 288] or VR headsets suffer from significant inaccuracies when there are occlusions and changes in lighting as found in Section 10.5. On the other hand, recent motion capture gloves using EMF sensors are significantly more accurate without being that expensive. They do not not suffer from occlusions like many vision-based techniques and return rich data about the skeleton joint-structure of the human hand. Furthermore, they can be worn comfortably on the human hand without impeding motion. In BiDex we choose the Manus Glove [247] as it has shown to provide accurate tracking readings, and does not overheat or face as many calibration issues as other alternatives such as the Rokoko gloves [326, 227]. However, controlling a robot hand from this human hand data is still challenging due to the morphology gap between these two hands. How can we map commands from the fingers of the human operator to the robot hand which can have a different kinematic structure?
If the robot hand kinematic structure is roughly human-like, one option would be to map joint angles directly between the human finger joints and robot hand joints. While these gloves do not have true joint angles, they can calculate joint

angles by an inverse kinematics solver onto a default human skeleton. However, if the robot fingers have different proportions and sizes to human fingers, then the motions will not match properly. The human thumb in particular has a complicated joint configuration that many robot hands do not perfectly emulate, making it very difficult to have intuitive thumb control. The pinch grasps will not be accurate which will adversely affects the ability to perform tasks reliably, since manipulation critically relies on the relative positions between fingertips.

Prior work [151, 369] have gotten around this issue by ensuring pinch grasp is consistent between the human hand and robot hands. Wang et al. [408] has shown that effective mapping can be achieved by optimizing for the joint position of the fingertip positions as well as the penultimate joint(DIP) positions on each finger with respect to the wrist to be similar between the human and robot hand using a SDLS IK solver. [45] We use the Manus gloves [247] with a similar inverse kinematics based approach since it enables precise pinch grasps and proper thumb placement.

Algorithm 8 Teleoperation Data System

```
Require: Two robot arms $Al, Ar$
Require: Two robot multi-fingered hands $Hl, Hr$
Require: Kinematic models for two arms $Kl$, $Kr$
Require: Gloves with fingertip trackers $Gl, Gr$
Require: IK model for robot fingertips $Q_{l,r}()$
Require: RGB workspace cameras $\{I_c\}$
Require: Number of trajectories to be collected $N$
 1: Initialize Data buffer $\mathcal{D}$
 2: for trajectory 1:N do
 3:     Initialize trajectory $\mathcal{T} = \{\}$
 4:     while task not completed do
 5:         Read camera images $\{I_c\}_t$
 6:         Read arm joints $Al_t, Ar_t$
 7:         Read robot hand joints $Hl_t$, $Hr_t$
 8:         Obs $o_t = \{Al_t, Ar_t, Hl_t, Hr_t, \{I_c\}_t\}$

 9:         Read kinematic arm model joints $Kl_t, Kr_t$
10:         Read glove fingertip positions $Gl_t$, $Gr_t$
11:         Finger joints $ql_t = Q_l(Gl_t), qr_t = Q_r(Gr_t)$
12:         Action $a_t = \{Kl_t, Kr_t, ql_t, qr_t\}$
13:         Add $(o_t, a_t) \mapsto \mathcal{T}$

14:         Set joints of $Al$ using $Kl_t$ and $Ar$ using $Kr_t$
15:         Set joints of $Hl$ using $ql_t$ and $Hr$ using $qr_t$
16:     end while
17:     Add $\mathcal{T} \mapsto \mathcal{D}$
18: end for
19: return Data buffer $\mathcal{D}$
```

9.3.2 Arm Tracking

Our system must track the human wrist pose accurately to control two robot arms. Traditionally, there are many approaches from both the motion capture community and robotics community alike. However, many of these approaches rely on calibrated external tracking devices which either are costly or have high latency. These external tracking devices are also non-portable, making it hard to scale to mobile systems. Instead, we leverage key insights from Zhao et al. [458], Wu et al.

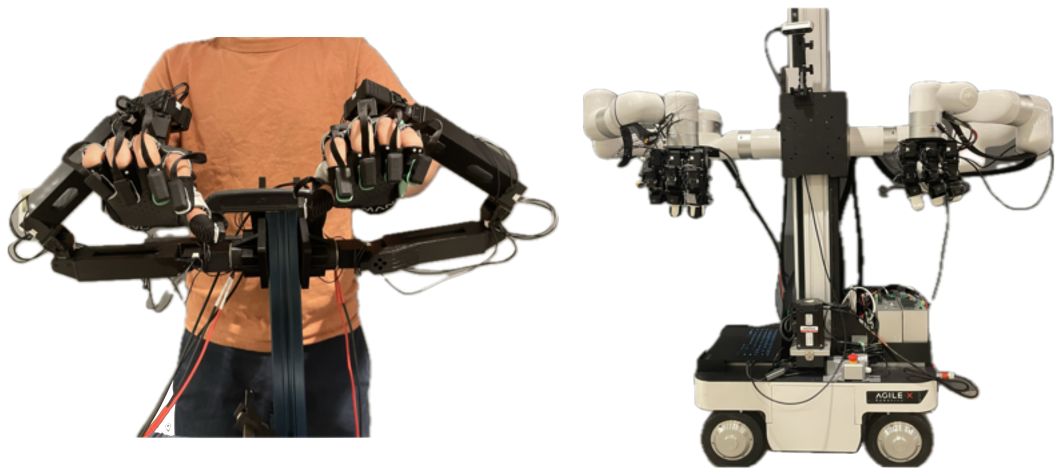


Figure 9.2: **Mobile bimanual teleoperation system.** Left: An operator strapped into BiDex.Right: Our bimanual robot setup including two xArm robot arms, two LEAP Hands [362] and three cameras on an AgileX base.

[426] which both use lighter teacher arms attached to the human arm to control a robot arm and hand system. Specifically we follow the GELLO system from Wu et al. [426] to teleoperate a full size robot arms. A key question is how to mount this arm-tracking system on a human wrist and hand. If the robot hand is mounted on the arm in a human-like way, then the glove needs to be mounted in a human-like way on the GELLO to match. However, this orientation means that the human arm will be parallel with the GELLO and constantly collide with it. This is jarring for the operator and uncomfortable. In BiDex, we mount the robot hands underneath the arms in the same orientation as if it were a gripper as in Figure 9.2. When mirroring this in the GELLO, the human arm and GELLO output are perpendicular to each other and do not collide which is more comfortable. Because of the weight of the motion capture glove, we must adjust the GELLO to be more robust. This includes adding a strong bearing to the base joint of the GELLO and adding rubber bands to bias additional joints back to the center of the joint range. We will release this CAD along with the rest of the code on our website to use this system.

9.3.3 Robot Configurations

Tabletop Manipulation: For our tabletop setup, the robotic arms are positioned to face each other, while the GELLO teaching arms mirror this configuration. Compared to a side-by-side configuration, this setup has three main benefits: 1) the human operators avoid collisions with the GELLO arms; 2) the setup allows better visibility of the workspace, which will be otherwise occluded by a side-by-side robot arm configuration; 3) and finally, the robot arms have a wider shared workspace.
Mobile Manipulation: BiDex does not require external tracking systems and is

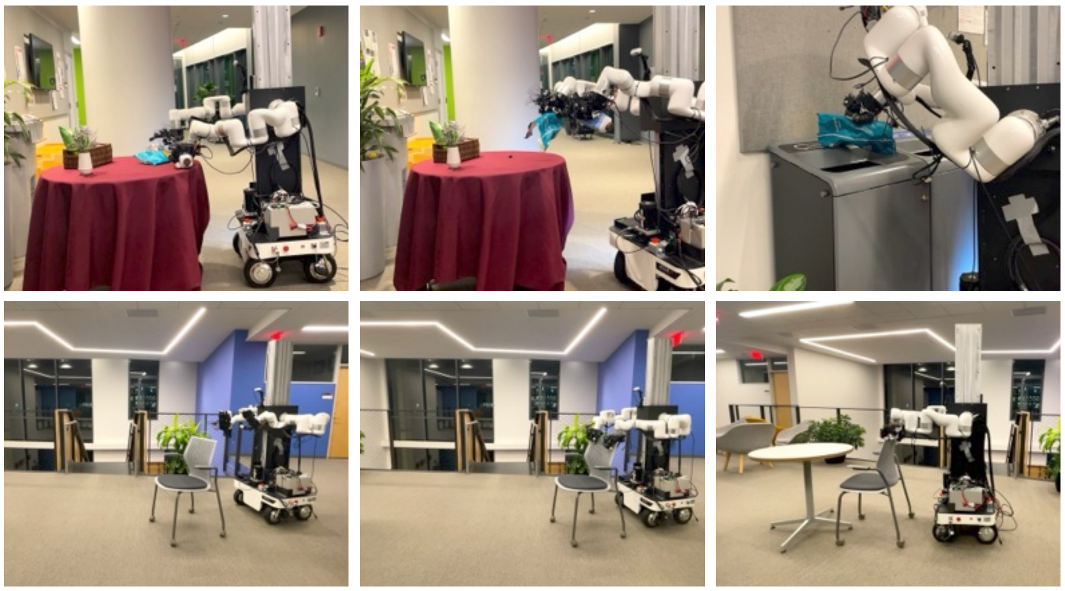


Figure 9.3: **All Tasks**: Teleoperation of the mobile robot systems with BiDex. **Top**: Picking up trash from a table and discarding it into a bin. **Bottom**: Grasping a chair and moving it to align with a table.

very lightweight so it can easily be used in the mobile setting. The teacher arms are instead mounted onto a compact mobile cart. Our cameras are all egocentric, one is mounted on the torso and two others are mounted on each of the wrists. For the mobile robot, we mount two robot arms onto an articulated torso that can move up towards high objects and down towards the ground similar to PR2 [34] as seen in Figure 9.2. The robotic assembly, including arms and torso, is mounted on an AgileX Ranger Mini enabling movement in any SE(2) direction required by the task [4]. A secondary operator manages the mobile base with a joystick, manages the task resets, and handles the data. Future implementations might include the addition of encoders, Dynamixel motors, or a SLAM camera to enhance tracking and maneuverability, inspired by Mobile Aloha [116].

9.4 Experiment Setup

9.4.1 Baseline Teleoperation Approaches

Vision based VR Headset: In recent years, the accessibility of low-cost VR headsets using multi-camera hand tracking has made them popular for teleoperation such as in [92, 282] As a baseline we use the Apple Vision Pro which returns both finger data similar to MANO parameters [288] and wrist coordinate frame data. The finger data is used in the same way as with BiDex through inverse kinematics-based retargeting and commanded onto the robot hands. The wrist data is reoriented, passed through

inverse kinematics and the final joint configuration is commanded to the arms.
SteamVR Tracking: SteamVR, commonly used in the video gaming community has also seen recent interest in the robotics community from industry [352] and academia alike. [246, 3] It uses active powered laser lighthouses that must be carefully placed around the perimeter of the workspace. Wearable pucks with IMUs and laser receivers are worn on the body of the operator. In our experiment the operator wears one tracker on each wrist and one tracker on their belly. The wrist location is calculated with respect to the belly pose, mapped to the robot arm and the joint angles are calculated using inverse kinematics. The hand tracking gloves are the same as BiDex.

9.4.2 Choice of Dexterous End-Effectors

Leap Hand: LEAP Hand, introduced by Shaw et al. [362] is a low-cost, easy-to-assemble robot hand with 16 DOF and 4 fingers. LEAP Hand introduces a novel joint configuration that optimizes for dexterity as well as human-like grasping. We use this hand for many experiments as it is a readily available dexterous hand available for comparison studies.
DLA Hand: We would like a hand that is smaller and more compliant than LEAP Hand. DLA Hand is crafted to mimic the suppleness and strength of the human hand with fingers that have a 3D-printed flexible outer skin paired with a sturdy inner framework resembling bones. These fingers do not break but instead bend and flex upon impact. We also introduce an active articulated palm which integrates two motorized joints, one spanning the fingers and another for the thumb, enabling natural tight grasping. DLA Hand contains 21 degrees of freedom and is sized to resemble a human hand, easy to assemble and is economical. Because of the human-like size and kinematics, it is easy to retarget to and can complete many more dexterous tasks successfully.

9.4.3 Task Descriptions

Tabletop: We show three tasks in the tabletop setting for all the teleoperation methods. In *Handover*, the robot picks up a pringles can and passes it from its right hand to its left hand in the air. In *Pour*, the robot pours from a glass bottle in one hand into a plastic cup held by the other hand. In *Tabletop Cup Stack* the agent stacks one cup into another cup in the other hand.
Mobile: The next three tasks we consider are in the mobile setting. In *Transport Box*, the agent moves a box from one table to another table using two hands. In *Mobile Chair Push*, the agent needs to grasp a chair, and then align it with a table. In *Clear Trash*, the robot clears trash from the table into a dustbin. We visualize the chair push and clear trash tasks in Figure 9.3.

	Completion Rate			Time Taken		
	Handover	Cup Stacking	Bottle Pouring	Handover	Cup Stacking	Bottle Pouring
`Vision Pro VR`	60	40	70	21.6	38.8	35.5
`SteamVR`	80	85	60	17.5	16.5	15.5
`BiDex`	95	75	85	6.5	15.5	14.9

Table 9.1: **Tabletop Teleoperation**: We compare BiDex on the handover, cup stacking, and bottle pouring tasks to two baseline methods, SteamVR and Vision Pro. BiDex enables more reliable and faster data collection, especially for harder tasks like bottle pouring.

9.5 Results

We investigate BiDex teleoperating in both the tabletop scenario and in the mobile in-the-wild scenario against a few baselines. For these comparisons, we use LEAP Hand by Shaw et al. [362] because it is an open source easy to assemble baseline robot hand that is readily attainable by any robotics lab. We also use BiDex with the more recent DLA Hand, which is made of a combination of rigid and soft material and capable of performing more complex tasks.

9.5.1 Bimanual Dexterous Teleoperation Results

BiDex provides more stable arm tracking: Because the teacher arm system is wired, BiDex is very reliable and does not fail. There is almost no jitter and very little latency which makes it ideal for arm tracking. The GELLO is very light and does not get in the way of the user more than the weight of the gloves. The kinematic feedback of arm resistance is very light but it helps the operator feel and work around arm singularities naturally. As seen in Table 9.1 and Table 9.2, BiDex achieves a higher completion rate while requiring less time for teleoperators. The Vision Pro often has jittery arm tracking which makes it difficult to teleoperate more difficult tasks. Low-pass filtering can help somewhat, but the latency added is undesirable. Occasionally the system will stop working completely which is jarring for the user. The SteamVR system is wireless so the user can be untethered and it is fairly accurate in its tracking. However, it can suffer from short periods of high latency or disconnections every 5-10 minutes that is jarring to the user. Significantly, we cannot use SteamVR outside in the mobile setting due to the external tracking lighthouses that must be setup around the teleoperation environment.
BiDex provides more accurate hand tracking: With BiDex, the fingertip tracking is very accurate with the Manus glove. When mapping to different robots, the inverse kinematics only has to be tuned slightly with each operator if their hand size is significantly different. The abduction-adduction of the MCP side joint is also very accurate in any condition. These advantages are especially apparently in DLA Hand

	Completion Rate			Time Taken		
	Chair Pushing	Box Carry	Clear Trash	Chair Pushing	Box Carry	Clear Trash
`Vision Pro VR`	75	75	50	15.0	33.7	79.8
`BiDex`	95	95	75	16.4	29.7	74.6

Table 9.2: **Mobile Teleoperation**: Completion rate and time taken averaged across 20 trials using a mobile bimanual system with LEAP Hand [362], for different tasks. BiDex is versatile and compact enough to be adopted to successfully collect data for mobile tasks.

where these accuracies are very important in doing more complex tasks.
With the Vision Pro, the hand size often changes slightly with different lighting conditions which makes it difficult to retarget to the robot hands. The abduction-adduction estimation for the fingers also changes with occlusions which makes it difficult to do more complex tasks. The latency is noticeable but this is not an issue when teleoperating for quasi-static tasks.

9.5.2 Training Dexterous Visuomotor Policies with BiDex

To ensure that the data that is collected by our system is high quality and useful for machine learning we train single task closed loop behavior cloning policies. Specifically, we train an action chunking transformer from [458] with a horizon length of 16 at 30hz using pretrained weights from [83] on around 50 demonstrations. The state space is the current joint angles of the robot hand and the images from the camera. The action space in the case of LEAP Hand is 16 dimensions for each hand and 6 dimensions for each arm for a total of 44 dimensions. During rollouts, the behavior of the policies are very smooth, exhibiting the high quality of the teleop data. In tasks such as the YCB Pringles can [47] handover, we even see good generalization of the policy to different initial locations of the can.

	Can Handover	Cup Stacking	Bottle Pouring
Leap Hand	7/10	14/20	16/20

Table 9.3: **Imitation learning**: We train ACT from [458] using data collected by BiDex and find that our system can perform well even in this 44 dimension action space. This demonstrates that our robot data is high quality for training robot policies.

9.5.3 Extreme Dexterity using DLA Hand

To push BiDex to its dexterous limits, we use DLA Hand: an extremely dexterous 21 DOF hybrid soft-rigid hand which is explained in Section 9.4.2. To do this, we show a variety of very challenging tasks as in Figure 9.1 and Figure 9.4. These tasks include pouring, scooping, hammering, chopstick picking, hanger picking, picking up basket, drilling, plate pickup and pot picking. In these experiments we find that BiDex scales well to this high DOF hand and it feels very natural to control this

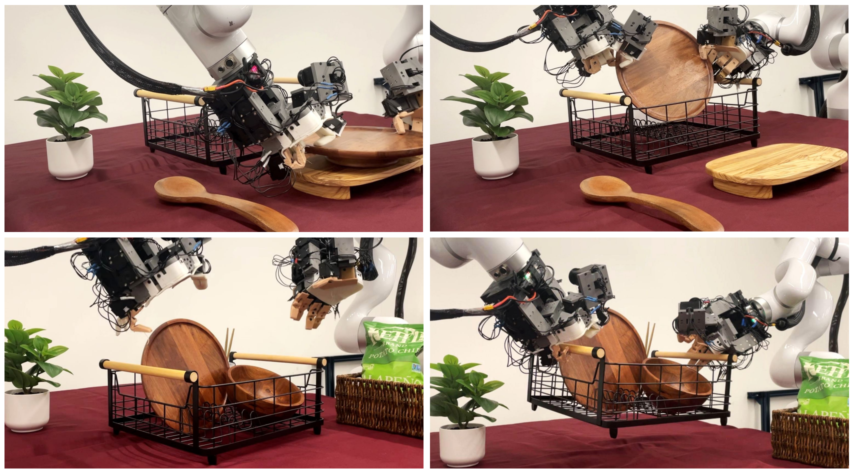


Figure 9.4: **Clearing the Dishes**: In this task, we use BiDex to perform a long horizon task to place bowls and spoons into a drying rack and lift the drying rack away from the table.

robot hand. We provide video results in the supplemental and on our website at https://bidex-teleop.github.io

9.6 Discussion and Limitations

In this Chapter, we introduce BiDex, a portable, low-cost and extremely accurate method for teleoperating a bimanual, human-like robot hand and arm system. We demonstrate the system's applicability to both a tabletop and a mobile setting and show its efficiency in performing bimanual dexterous tasks in comparison to alternative approaches including SteamVR and Vision Pro. Nevertheless, our BiDex is not without limitations. Due to the lack of haptic feedback, the human operator has to rely on visual feedback for teleoperation and cannot feel what the robot hand is feeling. Additionally, they cannot exert intricate force control and can only control the kinematics of the robot hand and arm which can make it challenging for fine-grained manipulation tasks. A promising direction in the future would be to integrate haptic feedback into our system which will unlock further potential for collecting extreme dexterity data.

Chapter 10

Neural MP: A Generalist Neural Motion Planner

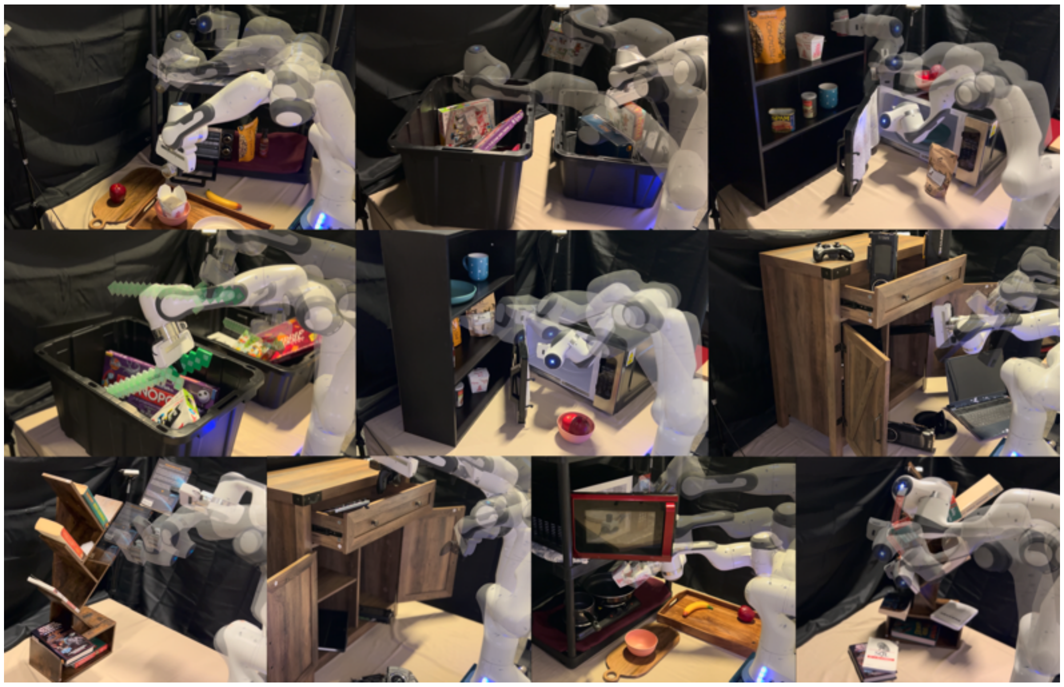


Figure 10.1: **Neural Motion Planning at Scale in the Real World.** We train a *single* neural network policy to solve motion planning problems across diverse tabletop setups enabling it to generate collision free motions for entirely unseen tasks such as placing books in bookcases, transferring items from shelves to microwaves, rearranging electronics from a drawer into a cabinet and moving irregular items between bins.

10.1 Motivation

Motion planning is a longstanding problem of interest in robotics, with previous approaches ranging from potential fields [191, 415, 305], sampling (RRTs and Roadmaps) [189, 215, 33, 185, 205, 120, 380], search (A*) [154, 226, 200] and trajectory optimization [318, 346, 95]. Despite being ubiquitous, these methods are slow at producing solutions since they largely plan from scratch at test time, re-using little to no information outside of the current problem and what is engineered by a human designer. On the other hand, humans can generate motions in a closed loop manner, move quickly, react to various dynamic obstacles, and generalize across a wide distribution of problem instances. Notably, this ability improves with experience. Consider an example of a person learning a new sport, like tennis. Initially, they might reason carefully about the form of their shots, but over time with repeated practice, they play faster, smoother and more effortlessly. Hence, practice *distills* planning into muscle memory. Can we enable such data-driven learning for robots to solve motion planning problems with ease?

The main challenge in training data-driven motion planning is the data collection itself, as scaling robotic data collection in real-world requires significant human time and effort. In recent years, there has been a concerted effort to scale up data collection for robot tasks [278, 192]. However, the level of diversity of scenes and arrangement of objects is still quite limited, especially for learning obstacle avoidance behavior needed to solve motion planning problems. Constructing such setups with diverse obstacle arrangements with numerous objects is prohibitively expensive in terms of cost and labor.

Instead, we look to leverage simulation, which has shown great promise in recent years in enabling policy learning for high-dof robots [218, 467, 61, 206, 144]. We build a large number of complex environments by combining procedural, programmatic generation with models of everyday objects sampled from large 3D datasets. These are used to collect expert data from state-of-the-art motion planners [380], which is then *distilled* into a reactive generalist policy. Because this policy has seen data from over **1 million** scenes, it is able to generalize to novel obstacles and scene configurations that it has never seen before. However, deploying policies in the real world might be unsafe for the system due to the potential of collisions. We mitigate this by using simple forward samplers to predict future states the robot will end up in, and run lightweight optimization to find a safe path.

Our core contribution is a simple, scalable approach to training and deploying fast, general purpose neural motion planners: 1) **large-scale procedural scene generation** with diverse environments in realistic configurations, 2) **multi-modal sequence modeling** for fitting to sampling-based motion planning data and 3) **lightweight test-time optimization** to ensure safe, reliable deployment in the real world. Furthermore, for each component, we propose concrete improvements to enable the overall method to perform well, namely an algorithm for producing diverse scenes via

Figure 10.2: **Visualization of Diverse Simulation Training Environments**: We train Neural MP on a wide array of motion planning problems generated in simulation, with significant pose, procedural asset, and mesh configuration randomization to enable generalization.

mixing programmatic assets and complex meshes, using simple, scalable and fast-to-deploy LSTM policies with Gaussian Mixture Models for fitting to multi-modal data, and test-time optimization via SDF-based collision checking of predicted policy trajectories. To our knowledge, Neural MP is the first to demonstrate that a neural policy trained in simulation can generalize to a broad set of out-of-distribution of real-world environments, generalizing across tasks with significant variation across poses, objects, obstacles, backgrounds, scene arrangements, in-hand objects, and start/goal pairs. We execute a first-of-its-kind real-world empirical study of motion-planning methods, evaluating our approach on **64** real world motion planning tasks across **four** diverse environments, demonstrating a motion planning success rate improvement of **23%** over sampling-based, **17%** over optimization-based and **80%** over state-of-the-art neural motion planning methods.

10.2 Related Work

Approaches for Training General-Purpose Robot Policies: Prior work on large scale imitation learning using expert demonstrations [37, 36, 278, 74, 365, 192] has shown that large models trained on large datasets can demonstrate strong performance on challenging tasks and some varying levels of generalization. On the other hand, sim2real transfer of RL policies trained with procedural scene generation has demonstrated strong capabilities for producing generalist robot policies in the locomotion regime [206, 467, 2, 61, 398]. In this work, we combine the strengths of these two approaches to produce powerful neural motion planning policies. We propose a method for procedural scene generation in simulation and combine it with large scale imitation learning to produce strong priors which we transfer directly to over 50 motion planning problems in the real world.

Procedural Scene Generation for robotics: Automatic scene generation and synthesis has been explored in vision and graphics [412, 53, 52, 325] while more recent work has focused on embodied AI and robotics settings [88, 413, 187, 74]. In particular, methods such as Robogen [413] and Gen2sim [187] use LLMs to propose tasks and build scenes using existing 3D model datasets [89] or text-to-3D [] and then decompose the tasks into components for RL, motion-planning and trajectory

optimization to solve and are limited to simulation only results. Our method is instead rule-based rather than LLM-based, is designed specifically for generating data to train neural motion planners (this involves answering questions such as how one should generate a diverse of poses to train such a planner - see Sec. 10.3.1), and demonstrates that policies trained on its data can indeed be transferred to the real world. MotionBenchmaker [51], on the other hand, is similar to our data generation method for motion planning in that it autonomously generates scenes using programmatic assets. However, the datasets generated by MotionBenchmaker do not contain significant realism: with floating robots, a single major obstacle per scene and primitive objects such as cuboids, spheres and cylinders that are spaced far apart. By comparison, the scenes and data generated by our work (Fig. 10.2) are considerably more diverse, containing additional programmatic assets that incorporate articulations (microwave, dishwasher), multiple large obstacles per scene (up to 5), complex meshes sampled from Objaverse [89], and tightly packed obstacles that are as close as possible.
Neural Motion Planning: Finally, there is a large body of recent work [306, 108, 307, 50, 169, 335, 167] focused on imitating motion planners in order to accelerate planning. MPNet [306, 180, 307] trains a network to imitate motion planners, then integrates this prior into a search procedure at test time. Our method trains a stronger prior by instead leveraging large scale scene generation and architectural improvements, allowing a simpler optimization process at test time while obtaining strong results across a diverse set of tasks. MπNets [108] trains the state of the art neural motion planning policy using procedural scene generation and demonstrates transfer to the real world. We adopt a similar approach, with 1) significantly more diverse scenes via programmatic asset generation and complex real-world meshes, 2) a more powerful learning architecture using sequence modeling and multi-modal Gaussian Mixture Model output distributions and 3) the incorporation of a test-time optimization technique to improve performance at deployment time.

10.3 Neural Motion Planning

Our approach enables generalist neural motion planners, by leveraging large amounts of training data generated in simulation via expert planners. The policies can generalize to out-of-distribution settings due to the diverse conditions and settings simulated, and using powerful deep learning architectures. To further improve the performance of these policies at deployment, we leverage test time optimization to select the best path out of a number of options. We now describe each of these pieces in more detail.

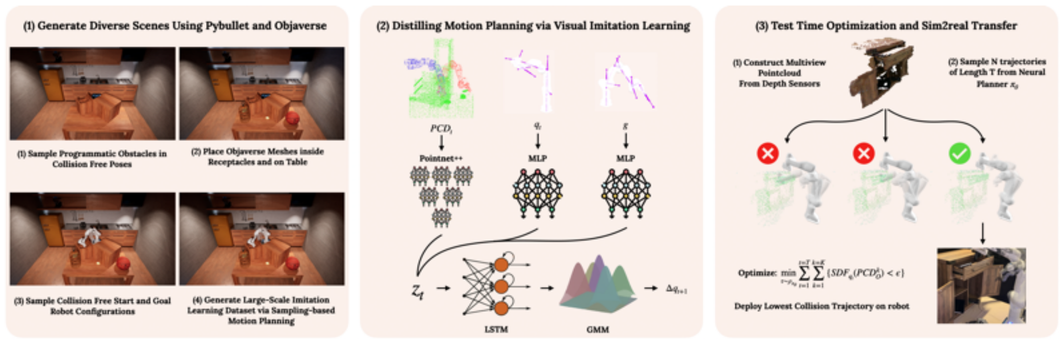

(1) Generate Diverse Scenes Using Pybullet and Objaverse
(1) Sample Programmatic Obstacles in Collision Free Poses
(2) Place Objaverse Meshes inside Receptacles and on Table
(3) Sample Collision Free Start and Goal Robot Configurations
(4) Generate Large-Scale Imitation Learning Dataset via Sampling-based Motion Planning
(2) Distilling Motion Planning via Visual Imitation Learning
PCD_t
q_t
g
Pointnet++
MLP
MLP
z_t
LSTM
GMM
Δq_{t+1}
(3) Test Time Optimization and Sim2real Transfer
(1) Construct Multiview Pointcloud From Depth Sensors
(2) Sample N trajectories of Length T from Neural Planner π_θ
Optimize: $\min_{\tau \sim \pi_\theta} \sum_{t=1}^{t=T} \sum_{k=1}^{k=K} (SDF_{q}(PCD_O^k) < \epsilon)$
Deploy Lowest Collision Trajectory on robot

Figure 10.2: **Method Overview**: We present Neural Motion Planners, which consists of 3 main components. **Left**: Large Scale data generation in simulation using expert planners **Middle**: Training deep network models to perform fast reactive motion planning **Right**: Test-time optimization at inference time to improve performance.

10.3.1 Large-scale Data Generation

One of the core lessons of the deep learning era is that the quality and quantity of data is crucial to train broadly capable models. We leverage simulation to generate very large datasets for training robot policies. Our approach generates assets using programmatic generation of primitives and by sampling from diverse meshes of everyday objects. These assets are then combined to create complex scenes resembling real world scenarios the robot might encounter (Fig. 10.2). , as described in Alg. 9.
Procedural generation from primitives: How do we generate a large enough number of diverse environments to train a generalist policy? Hand designing each environment is tedious, requiring significant human effort per scene, which doesn't scale well. Instead, we take the approach of *procedural* scene generation, using a set of six *parametrically variable* categories - shelves, cubbies, microwaves, dishwashers, open boxes, and cabinets. These categories are representative of a large set of objects in everyday scenarios that robots encounter and have to avoid colliding with. Each category instance is constructed using a combination of primitive cuboid objects and is parameterized by category specific parameters which define the asset. Specifically a category instance g is comprised of N cuboids $g = \{x_0..x_i...x_N\}$, which satisfy the category level constraint given by $C(g)$. For controlled variation within each category, we make use of *parametric* category specific generation functions $X(\mathbf{p}) = \{x_0..x_i.x_N\}$, s.t. $C(X(\mathbf{p}))$, where p specifies the size and scale of each of the cuboids, their relative positions, and specific axes of articulation. The constraint C(.) relates to the relative positions, scales and orientations of the different cuboids, *e.g* for the microwave category the constraint ensures each of the walls are of the same height, and that the microwave has a hinge door.
Objaverse assets for everyday objects: While programmatic generation can create a large number of scenes using the defined categories, there are a large number of everyday objects the robot might encounter that lie outside this distribution. For

example, a robot will need to avoid collisions with potted plants, bowls and utensils while moving between locations, as shown in Fig .10.1. To better handle these settings, we augment our dataset with objects sampled from the recently proposed large-scale 3D object dataset, Objaverse [89]. This dataset contains a wide variety of objects that the neural planner is likely to observe during deployment, such as comic books, jars, record players, caps, etc. We sample these Objaverse assets in the task-relevant sampling location of the programmatic asset(s) in the scene, such as between shelf rungs, inside cubbies or within cabinets.

Complex scene generation: The scenes we use comprise combinations of the procedurally generated assets built from primitives, and the Objaverse assets arranged on a flat tabletop surface. A naive approach to constructing realistic scenes is to use rejection-sampling based on collision. This involves iteratively sampling assets on a surface, and resampling those that collide with the current environment. However, as the number, size and type of objects increases, so does the probability of sampling assets that are in collision, making such a process prohibitively expensive to produce a valid configuration. In addition, this is biased towards simple scenes with few assets that are less likely to collide, which is not ideal for training generalist policies. Instead, we propose an approach that iteratively adds assets to a scene by adjusting their position using the effective collision normal vector, computed from the existing assets in the scene. Please see Alg. 9 for the full procedure and the Appendix for additional details.

Algorithm 9 Complex scene generation

```
Require: Asset category generators $\{X_i(\mathbf{p})\}_{0,1..G}$
Require: Number of scenes $N$
Require: Max objects per scene $K$
Require: Collision checker $Q$
 1: for scene 1: N do
 2:     Initialize scene $S = \{\}$
 3:     Sample number of assets $k \sim [1, ...K]$
 4:     for asset 1:k do
 5:         Sample asset category $g \sim [0, ..N]$
 6:         Sample asset parameter $p$
 7:         Sample asset $x \sim X_g(p)$
 8:         while $Q(S, x)$ do
 9:             for each asset $s_i$ in $S$ do
10:                 $n_i$ = collision normal b/n $x$ and $s_i$
11:             end for
12:             Effective collision normal $n = \sum n_i$
13:             Update $p$ so $X_g(p)$ center is shifted along $n$
14:         end while
15:         Add asset $x$ to scene $S$
16:     end for
17:     yield scene S
18: end for
```

Motion Planning Experts: To collect expert data in the diverse generated scenes, we leverage state-of-the-art sampling-based motion planners due to their (relative) speed as well as ease of application to a wide array of tasks. Specifically, we use Adaptively Informed Trees [380] (AIT*), an almost-surely asymptotically optimal sampling-based planner to produce high-quality plans using privileged information, namely access to a perfect collision checker in simulation. How do we ensure that the planner is evaluated between points in the scene that require it to maneuver around obstacles? We generate tight-space configurations by sampling end-effector poses from specific locations (*e.g.*, inside a cubby or microwave) and by using inverse kinematics (IK) to derive the joint pose. Tight-space configurations are sampled 50% of the time, to ensure that we collect trajectories where the robot moves around obstacles, as opposed to taking straight line paths between nearby free space points. Additionally, we spawn objects grasped in the end-effectors, with significant randomization including boxes, cylinders, spheres or even Objaverse meshes. Importantly, we found that naively imitating the output of the planner performs poorly in practice as the planner output is not well suited for learning. Specifically, plans produced by AIT* often result in way-points that are far apart, creating large action jumps and sparse data coverage, making it difficult for networks to fit the data. To address this issue, we perform smoothing using cubic spline interpolation while enforcing velocity and acceleration limits. We found that smoothing is crucial for learning performance as it ensures action size limits for each time-step transition.

10.3.2 Generalist Neural Policies

We would like to obtain agents that can use diverse sets of experiences to plan efficiently in new settings. In order to build generalist neural motion planning policies, we need an observation space amenable to sim2real transfer, and utilize an architecture capable of absorbing vast amounts of data.

Observations: We begin by addressing the sim2real transfer problem, which requires considering the observation and action spaces of the trained policy. With regards to observation, point-clouds are a natural representation of the scene for transfer, as they are 3D points grounded in the base frame of the robot and hence view agnostic, and are largely consistent between sim and real. This has also been demonstrated in prior work for sim2real transfer in a variety of tasks using point-clouds [108, 68, 179, 58, 423]. We include proprioceptive and goal information in the observations, consisting of the current joint angles q_t, the target joint angles g, in addition to the point-cloud PCD.

Network Architecture: We utilize an architecture capable of scaling with data while performing well on a multi-modal sequential control problem such as motion planning. To that end, we design our policy π (visualized in Fig. 10.2) to be a sequence model to imitate the expert using a notion of history which is useful for fitting privileged experts using partially observed data [74]. In principle, any sequence modeling architecture could be used, but in this work, we opt for LSTMs for their fast inference time and comparable performance to Transformers on our datasets (see Appendix). We operate the LSTM policy over joint embeddings of PCD_t, q_t, and g with a history length of 2. We encode point-clouds using PointNet++ [301], which is an efficient and effective point-cloud processing architecture that performs well in practice, while we use MLPs to encode q_t and g_t. For each timestep, we concatenate the embeddings of each of the observations together into one vector and then pass them into the LSTM for action prediction. For the output of the model, note that sampling-based motion planners such as AIT* are heavily multi-modal: for the same scene they may give entirely different plans for different runs. As a result, our policy requires an expressive, multi-modal distribution to effectively capture such data, for which we use a Gaussian Mixture Model (GMM) distribution for the policy. Specifically, Neural MP predicts a distribution over delta joint angles (Δq_{t+1}), which are used to compute the next target joint waypoint during deployment: $q_{t+1} = q_t + \Delta q_{t+1}$. We minimize the negative log-likelihood of the GMM distribution, which outperforms the PointMatch loss from Fishman et al. [108] (Sec. 10.5).

10.3.3 Deploying Neural Motion Planners

Test time Optimization: While our base neural policy is quite capable of solving a wide array of challenging motion planning problems, we would still like to ensure that these motions are safe to be deployed in real environments. We enable this property by combining our learned policy with a simple light-weight optimization procedure at inference time. This relies on a simple model that assumes the obstacles do not move and the controller can accurately reach the target way-points. Hence, given state s, the predicted state $s' = s + \hat{a}$ where $\hat{a}$ is the policy prediction. With this forward model, we can sample N trajectories from the policy using the initial scene point-cloud to provide the obstacle representation and estimate the number of scene points that intersect the robot using the linear forward model. We then optimize for the path with the least robot-scene intersection in the environment, using the robot Signed Distance Function (SDF). Specifically, we optimize the following objective at test time:

$$\min_{\tau \sim \rho_{\pi_\theta}} \sum_{t=1}^{t=T} \sum_{k=1}^{k=K} \mathbb{1}\{SDF_{q_t}(PCD_O^k) < \epsilon\}, T = 100, K = 4096, \epsilon = 0.01 \tag{10.1}$$

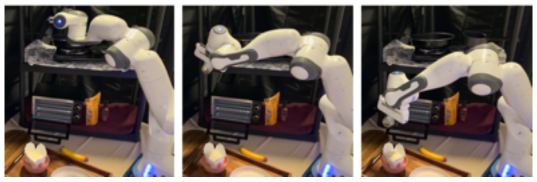
in which ρ_{π_θ} is the distribution of trajectories under policy π_θ with a linear model as described above, PCD_O^k is the kth point of the obstacle point-cloud and SDF_{q_t} is the SDF of the robot at the current joint angles. In practice, we optimize this objective with finite samples in a single step, computing the with minimal objective value by selecting the path with minimal objective value across 100 trajectories. We include a detailed analysis of the properties of our proposed test-time optimization approach in the Appendix.

Sim2real and Deployment: For executing our method on a real robot, we predict delta joint way-points which we then linearly interpolate and execute using a joint space controller. Our setup includes four extrinsically calibrated Intel RealSense cameras (two 435 and two 435i) positioned at the table's corners. To produce the segmented point cloud for input to the robot, we compute a point-cloud of the scene using the 4 cameras, segment out the partial robot cloud using a mesh-based representation of the robot to exclude points. We then generate the current robot and target robot point clouds using forward kinematics on the mesh-based representation of the robot and place them into the scene. For real-world vision-based collision checking, we calculate the SDF between the point cloud and the spherical representation of the robot, enabling fast SDF calculation (0.01-0.02s per query), though this method can lack precision for tight spaces.

10.4 Experimental Setup

In our experiments, we consider motion planning in four different real world environments containing obstacles (see Appendix). Importantly, these are not included as part of the training set, and hence the policy needs to generalize to perform well on these settings. We begin by describing our environment design, then each of the environments in detail, and finally our evaluation protocol and comparisons.

(a) Sampling-based planners struggle with tight spaces, a regime in which Neural MP performs well.


(b) Our method is able to motion plan with objects in-hand, a crucial skill for manipulation.

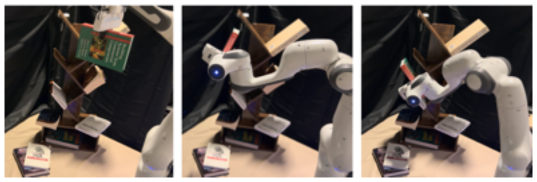

(c) Our policy has not been trained on this bookcase, yet it is able to insert the book into the correct location.

Figure 10.3: Emergent Capabilities

Environment Design: We evaluate our motion planner on tabletop motion planning tasks which we subdivide into *environments*, *scenes*, and *configurations*. We evaluate on four different environments, with each environment containing 1-2 large receptacles that function as the primary obstacles. For each environment, we have four different scenes which involve significant pose variation (over the entire tabletop) of the primary obstacles, table height randomization, as well as randomized selection, pose and orientation of objects contained within the receptacles. For each environment, we have two scenes with obstacles and two without obstacles. For each scene, we evaluate on four different types of start (q_0) and goal (g) angle pairs: 1) free space to free space, 2) free space to tight space 3) tight space to free space 4) tight space to tight space. Free space configurations do not have an obstacle in the vicinity of the end-effector, while tight space configurations generally have obstacles on most sides of the end-effector.

Our four environments are 1) **Bins**: moving in-between, around and inside two different industrial bins 2) **Shelf**: moving in-between and around the rungs of a

black shelf 3) **Articulated**: moving inside and within cubbies, drawers and doors 4) **In-hand**: moving between rungs of a shelf while holding different objects. For detailed descriptions of environment setups, we refer the reader to the Appendix.
Evaluation Protocol: We evaluate all methods on open loop planning performance for fairness, though our method, just like MπNets, is capable of executing trajectories in a closed loop manner. For neural planners such as our method and MπNets, this involves generating an open loop path by passing the agent's predictions back into itself using a linear model for the next state, as described in Sec. 10.3.3. We then execute the plans on the robot, recording the success rate of the robot in reaching the goal, its collision rate and the time taken to reach the goal. We follow MπNets' definition of success rate: reaching within 1cm and 15 degrees of the goal end-effector pose of the target goal configuration while also not colliding with anything in the scene. In practice, our policies achieve orientation errors significantly below this threshold, 2 degrees or less.
Comparisons: We propose three primary baselines for comparisons, that evaluate different aspects of our methods capability. We compare against sampling-based motion planning in the real world, which is expensive to run but has strong guarantees on performance. The first baseline is the expert we use to train our model, AIT* with 80 seconds of planning time. We run this planner with the same vision-based collision checker used by our method in the real world. AIT*-80s is extremely expensive and impractical to deploy in most settings due to its significant planning time. Hence, we additionally compare to AIT* with 10 seconds of planning time, which uses comparable planning time to our method (4s). We use AIT* with 10s of planning time as with 4s of planning time it was unable to find a plan for any real-world task. Next, we compare against Curobo [385], a state-of-the-art motion-generation method which performs GPU-parallelized optimization and is orders of magnitude faster than AIT*. We run this baseline with a voxel-based collision checker and optimize its voxel resolution per task due to its sensitivity to that parameter. Finally, we compare against the state-of-the-art neural motion planning approach, MπNets, which operates over the same visual point-cloud input as our method.

10.5 Experimental Results

To guide our experimental evaluation, we pose a set of experimental questions. 1) Can a single neural policy trained in simulation learn to solve complex motion planning problems in the real world? 2) How does our neural planner compare to state-of-the-art neural planning, sampling-based and trajectory optimization planning approaches? 3) How well does Neural MP extend to motion planning tasks with objects in hand? 4) What are the impacts key ingredients of Neural MP have on its performance?
Free Hand Motion Planning: In this set of experiments, we evaluate motion planning when there is no object in the robot's hand. The base policy on its own performs

well, achieving performance close to AIT*-80s with only 1 second of planning time. When we include test-time optimization, we find that across all three tasks, Neural MP achieves the best performance with a 95.83% success rate. In general, we find that Bins is the easiest task, with the sampling-based methods performing well, Shelf is a bit more difficult as it requires simultaneous vertical and horizontal collision avoidance, while Articulated is the most challenging task as it contains a diverse set of obstacles and tight spaces. Neural MP performs well across each task as it is trained with a diverse set of parametric objects that cover the types of real-world obstacles we encounter and it also incorporates complex meshes which cover the irregular geometries of the additional objects we include.

	Bins	Shelf	Articulated	Average
Sampling-based Planning:				
`AIT*-80s` [380]	93.75	75.0	50.0	72.92
`AIT*-10s (fast)` [380]	75.0	37.5	25.0	45.83
Optimization-based Planning:				
`Curobo` [385]	93.75	81.25	62.5	79.17
Neural:				
`MπNets` [108]	18.75	25.0	6.25	16.67
`Ours-Base Policy`	81.25	75.0	43.75	66.67
`Ours`	**100**	**100**	**87.5**	**95.83**

Figure 10.4: Neural MP performs best across each scene free-hand motion planning task, demonstrating greater improvement as the task complexity grows.

MπNets performed poorly on our tasks across the board. We attribute this finding to the fact that MπNets is only trained on data in which the expert goes from tight spaces to tight-spaces, which means the network has not observed joint angles that are in-free space and so it does not generalize well to such scenarios and the end-effector point matching loss in MπNets fails to distinguish between 0 and 180 degree rotations of the end-effector, so the network has not learned how to match the target end-effector pose when there is ambiguity. If we change the success rate metric for MπNets to count 180 degree flipped end-effector poses as successes as well, the average success rate of MπNets improves by 12.5% to 29.17%, yet it is still far below the other methods. Meanwhile failure cases for AIT* and Curobo were tight spaces for which vision-based collision checking is inaccurate and the probability of sampling/optimizing for a valid path is low. Our method performs well in practice, generalizing to 48 different unseen environment, scene, obstacle and joint configuration combinations, outperforming both versions of AIT*(80s and 10s) as well as MπNets by 23%, 50% and 79% respectively.

In Hand Motion Planning: In this experiment, we extend our evaluation to settings which are more task relevant: motion planning with objects in hand. We evaluate Neural MP against running the neural policy without test time optimization and without including any Objaverse data, achieving 81% performance vs. 31% and 44%. We visualize an example trajectory in Fig. 10.3. Our method performs well on in-distribution objects such as the book and board game, but struggles on more out of distribution objects such as the toy sword, which is double the size of objects at training time. We additionally deploy our method on significantly out of distribution objects such as the bookcase, and find that Neural MP generalizes well, capably inserting the book at the desired pose. This experiment also serves as an ablation of our method, demonstrating the utility of test time optimization on challenging out of distribution scenarios, which improves the performance of our method by a wide margin, 50%, on in hand motion planning. For this task, the base policy performance results in a large number of collisions as two of the in-hand objects are out of distribution (sword and board game), but the optimization step is able to largely remove them and produce clean behavior that reaches the target without colliding. In addition, we find that the Objaverse data is crucial for the success of our method on this task. The task setup contains a large number of diverse objects, which is not present in the dataset without Objaverse objects. As a result, a model trained only on the parametric assets consisting of cuboids will not generalize well to such complex meshes. This experiment confirms the utility of including complex meshes when generating scenes for distilling motion planning.

Comparisons to Learning-based Motion Planners: We next evaluate how Neural MP compares to two additional learning-based methods, MPNets [306] and EDMP [335] as well as MπNets [108] in simulation. We compare these three neural motion planning methods in simulation trained on the same dataset (from MπNets) of 3.27 million trajectories. We train policies on the Global expert data and the Hybrid datasets and then evaluate on 5400 test problems across the Global, Hybrid and Both solvable subsets. We include numerical results Tab. 10.5, with numbers for the baselines taken from the EDMP and MπNets papers. We find that across the board, Neural MP is the best learning-based motion planning method, outperforming both EDMP and MπNets on the test tasks provided in the MπNets paper. We attribute this to the use of sequence modelling in the form the RNN, the ability of the GMM to fit multimodal data and test-time optimization.

	Global	Hybrid	Both	Average
MPNet [306]				
Hybrid Expert	41.33	65.28	67.67	58.09
MπNets [108]				
Global Expert	75.06	80.39	82.78	79.41
Hybrid Expert	75.78	95.33	95.06	88.72
EDMP [335]				
Global Expert	71.67	82.84	82.79	79.10
Hybrid Expert	75.93	86.13	85.06	82.37
Ours				
Global Expert	**77.93**	85.50	87.67	83.70
Hybrid Expert	76.33	**97.28**	**96.78**	**90.13**

Figure 10.5: Performance comparison of neural motion planning methods across 5400 test problems in the MπNets dataset in simulation. Neural MP achieves the state-of-the-art results on these tasks.

Data Scaling: We evaluate the scaling of our method with data in order to understand how performance changes with dataset size. To do so, we train 3 additional models, with 1K trajectories, 10K trajectories and 100K trajectories. In these experiments, we train with subsets of our overall dataset and evaluate on held out simulation environments which are not sampled from the training distribution. While performance with a thousand trajectories is weak (15%), we find rapid improvement as we increase the orders of magnitude of data, with the model trained on 1M trajectories achieving 80% success rate on entirely held out tight-space shelf and bin configurations, showing that our method scales and improves with data.

Additional Ablations: We run ablations of components of our method (training objective, observation composition) in simulation to evaluate which have the most impact. For each ablation we evaluate performance on held out scenes. For training objective, we find that GMM (ours) outperforms L2 loss, L1 loss, and PointMatch Loss (MπNets) by (7%, 12%, and 24%) respectively. We find that including both q and g vectors is crucial for performance as we observe a 62%, 65%, and 75% performance drop when using only g, only q and neither q nor g respectively. We refer the reader to the Appendix for further analysis, discussion and results.

10.6 Discussion and Limitations

In this work, we present Neural MP, a method that builds a data-driven policy for motion planning by scaling procedural scene generation, distilling sampling-based motion planning and improving at test-time via refinement. Our model demonstrably improves over the sampling-based planning in the real world, operating 2.5x-20x faster than AIT* while improving by over 20% in terms of motion planning success rate. Notably, our model generalizes to a wide distribution of task instances and demonstrates favorable scaling properties. At the same time, there is significant room for future work to improve upon, our model 1) is susceptible to point-cloud quality, which may require improving 3D representations via implicit models such as NeRFs [255], 2) does not still handle tight spaces well, a capability which could be potentially acquired via RL fine-tuning of the base policy and 3) is slower than simply running the policy directly due to test-time optimization, which can be addressed by leveraging learned collision checking [262, 79].

Chapter 11

Conclusions, Discussion and Future Work

11.1 Summary

In this thesis, we discussed how to enable exploration for continually improving robots. We first consider exploration objectives, even in the absence of rewards or demonstrations. In Chapter 2, we show an explorer-achiever framework that uses a world model to estimate the uncertainty of action sequences, and practices reaching goals in imagination. We further improve sample efficiency by decoupling environment and agent-centric objectives in Chapter 3 for real robot deployment, prioritizing actions that lead to changes in the visual features of objects. Next, we ask how to build generalist explorers using prior video. In Chapter 4, we present an approach for representing affordances from human videos that focus on where in the scene to interact, and how to move after contact. We utilize this shared action space in Chapter 5 to build joint human-robot world models, that can perform multiple tasks. For future instantiations of world models, we look to video diffusion models, and in Chapter 6 study efficient adaptation of these models using reward gradient information. The third question we ask is how to enable greater autonomy via mobile manipulation systems. In Chapter 7, we present a real-world RL framework that enables a Spot robot with an arm to autonomously learn skills like sweeping with a broom or moving chairs, via task-relevant autonomy, policy learning that leverages behavior priors, and flexible reward specification using segmentation and detection models. We build our own custom low-cost mobile manipulator system in Chapter 8 for operating various articulated objects such as doors, drawers, cabinets across campus, using interactive online learning. Finally, we study approaches for scaling data collection for enabling stronger initial policies for deployment. In Chapter 9, we build a low-cost teleoperation system for a bimanual dexterous system with robot arms and multi-fingered hands, by combining kinematic link structure models for arm tracking along with a motion capture fingertip glove for tracking finger movements accurately and reliably. In Chapter 10, we use simulation for large scale procedural scene generation for training neural motion planners, which can generalize to many out of distribution real world scenarios.

11.2 Discussion, Lessons Learned and Future Work

Exploration as a combination of top-down and bottom-up processes: Over the course of my work in exploration, a major theme has been improving sample efficiency. This has been important for real-world improvement, since it enables the agent to learn skills in a reasonable time frame. My earlier work started out using purely *bottom-up* exploration [250], which utilized statistical uncertainty to drive actions. While this is a powerful framework for discovering new skills, the large number of samples required precludes real world deployment. For faster learning I have incorporated *top-down* conditioning, which provides useful priors on what actions are likely to provide useful learning signal. I have experimented with this in different forms, gradually adding stronger forms of top-down signal for exploration. First, we investigated the idea of *environment change*, which incentivizes actions that cause changes in the visual features of objects [251]. This leverages the prior that object interactions are beneficial for learning manipulation skills, and is not driven purely by bottom-up statistical considerations. Next, we sought to learn data-driven representations for an efficient exploration action space, using human videos [20, 252]. Here the top-down signal is stronger, since it communicates entire trajectory information of how objects are likely to be manipulated by humans. For mobile manipulation, we used task-specific action priors for our work that learn to operate articulated objects [432], inspired by how humans open doors. We also presented a more general framework for learning multiple tasks, where we use pre-existing navigation and grasping skills along with segmentation and detection models to maintain control over an object of interest before attempting to learn policy behavior [253]. Furthermore, we utilize behavior priors that provide better learning signal when training policies. Across all these different works, we see the common thread of injecting top-down signal for structuring the exploration space, to make improvement via online learning feasible in a short time-span.
Going forward, a more general way to incorporate this top-down information for directing exploration for autonomous agents might be to leverage powerful, general-purpose language and vision foundation models (LLMs/VLMs) [40, 392]. These models have been shown to have impressive reasoning capabilities and can provide a lot of prior information that is useful for robotics. There is already exciting work that demonstrates their application for autonomous agents in simulated domains to explore and learn skills for games such as minecraft [409]. So far, their application for robotics has been limited due to lack of sufficiently strong spatial reasoning for images, fine-grained understanding of object geometry and interaction dynamics, and consistent identification of different objects across time. However, these short-

comings will likely be addressed in future iterations, especially if these models also incorporate video training, which contains important multi-frame temporal information. It is important to reiterate that bottom-up signal is still crucial for discovering and learning robot skills, since a large variety of tasks require low-level control which LLMs/VLMs are not directly trained for. Furthermore, bottom-up signal driven purely by environment interaction allows discovering new behaviors not represented by top-down priors. Recent work has sought to directly integrate these by proposing Vision-Language-Action models [193], that utilize internet-trained representations from language/vision and are fine-tuned for action prediction. Future formulations that better preserve semantic reasoning capabilities of LLMs/VLMs and can also incorporate low-level control, while allowing for rapid online learning of new skills present promising directions of study.

Contextualizing exploration in the current robot learning landscape: The guiding goal of this thesis has been to enable robots to improve their capabilities via interaction, and we have demonstrated promising results along this direction. We are currently in a very exciting time for machine learning, with scaling data and compute demonstrating remarkable results in the fields of language, vision and video [40, 392, 314, 39], to the degree that models have begun to be deployed for real-world use cases. There is now concerted effort in industry as well as academia to replicate this success of data-driven learning for robotics. While I would like the systems we deploy to ultimately keep improving in capability via exploration, there is currently a different immediate priority. This involves building policies that are powerful enough to be deployed in some limited settings, for performing useful tasks. My view of moving towards a future with general-purpose robots now involves a bootstrapping process, where we first get data-driven policies that are quite capable, albeit for a restricted class of tasks, to the point that they can be deployed. Once policies begin to be deployed, and the robot fleet size increases, and robot systems are engineered to operate autonomously for long hours with minimal human intervention, then improvement via exploration for discovery will enable these robots to expand their range of skills. This view is driven by the need for data to drive learning, for without a sufficiently large number of robots that autonomously practice and explore, engineering effort is better invested in enabling competent initial policies that will be instrumental in increasing deployment fleet size. Eventual large scale interactive learning will also require careful system orchestration, involving safety constraints for real world sampling, reward specification, real time updates of the model and transfer of newly collected robot data. There are currently two predominant approaches for enabling strong initial deployment policies, both of which we have touched upon in the final section of the thesis - imitation learning and large-scale simulation training. There is currently the exciting possibility that larger scale execution of these ideas with greater compute and hardware manufacturing investments coupled with scalable data modelling techniques from vision/language domains will yield robots that can be deployed, setting off a data flywheel.

Simulation as a playground for emergent discovery: Of the aforementioned approaches for enabling the first version of deployment policies, I am particularly optimistic about simulation, specifically when combined with reinforcement learning. This has already shown amazing results for learning policies for locomotion for quadrupeds [206, 2] and humanoids [310], bipeds to learn how to play soccer [144], drones to navigate challenging courses [188], and in-hand manipulation [152, 6]. The application to quadruped locomotion has now even entered mainstream deployment in industry. While tackling open-world manipulation using simulation remains an open challenge, utilizing reinforcement learning with domain randomization holds the promise of discovering very robust high-frequency reactive policies, which is crucial for generalizing to various perturbations during deployment and dealing with environment uncertainty. Furthermore, with sufficiently powerful simulations that depict various scenes and environments with high degree of complexity, we can deploy exploration algorithms in simulators to discover skills for agents. Key ideas we have discussed of formulating objectives, learning from videos, utilizing priors to shape exploration can all be investigated now with a massive virtual fleet of robots. Hence, we can revisit our ideas of discovery and continually expanding repertoire of skills, but without the high cost of direct real world interaction. However, for such policies to be transferable to the real world, it is likely that simulators would have to very closely resemble the real world in terms of interaction physics and photorealism. There is some argument on whether domain randomization with approximate modelling is sufficient, but even this would require a very strong prior on the approximate feasible range of physics parameters. Most current simulators utilize hand-written rules and diverge from real-world physics in significant ways, hence mitigating this gap will be important.
World models as data-driven sims for generalist control: To solve the simulator modelling problem, we can resort to the tried and tested approach of utilizing data-driven learning. This has already been discussed in this thesis in the form of world models [140, 145], which predict future outcomes in a learned latent space (Chapters 2, 3, 5). These world models were trained for a particular environment (using human video pretraining in Chapter 5), but generalist simulators will need to utilize all the data available. A promising direction for this is to build off work in video modelling, which captures some important conceptual aspects and physics information present in videos [39], and we presented an approach for adapting such internet-pretrained models using reward information in Chapter 6. However, these video models currently also often hallucinate and produce generations that do not obey real-world physics. Nevertheless, this presents an interesting avenue for research. There have also been alternate formulations for world models, which learn representations that are not grounded in visual pixel reconstruction [14, 122]. A data-driven simulator that can predict outcomes in many different environments for various action sequences can then be used to learn skills with RL.

Chapter L

Appendix

L.1 Discovering and Achieving Goals via World Models

L.1.1 Experimental Details

Environments: The episode length is 150 for RoboBin and RoboKitchen and 1000 for RoboYoga. We show all goals in Figure L.1. For both *Walker* and *Quadruped*, the success criterion is based on the largest violation across all joints. The global rotation of the Quadruped is expressed as the three independent Euler angles. Global position is not taken into account for the success computation. *RoboBin*. The success criterion is based on placing all objects in the correct position within 10 cm. For reaching task, the success is based on placing the arm in the correct position within 10 cm. *RoboKitchen* uses 6 degrees of freedom end-effector control implemented with simulation-based inverse kinematics. The success criterion is based on placing all objects in the correct position with a threshold manually determined by visual inspection. Note that this is a strict criterion: the robot needs to place the object in the correct position, while not perturbing any other objects.

Evaluation: We reported success percentage at the final step of the episode. All experiments on our benchmark as well as on the SkewFit benchmark were ran 3 seeds. Due to large required compute, DISCERN and Plan2Explore results for LEXA were only run with one seed. The DISCERN and Plan2Explore results should therefore not be used for rigorous comparisons, but are nevertheless indicative of the simplicity of these benchmarks. Plots were produced by binning every 3e5 samples. Heatmap shows performance at the best timestep. Each model was trained on a single high-end GPU provided by either an internal cluster or a cloud provider. The training took 2 to 5 days. The final experiments required approximately 100 training runs, totalling approximately 200 GPU-days of used resources.

Implementation: We base our agent on the Dreamer implementation. For sampling goals to train the achiever, we sample a batch of replay buffer trajectories and sample both the initial and the goal state from the same batch, therefore creating a mix of easy and hard goals. To collect data in the real environment with the achiever, we sample the goal uniformly from the replay buffer. We include code in the supplementary material. The code to reproduce all experiments is available at `https://github.com/orybkin/lexa`

Hyperparameters: LEXA hyperparameters follow Dreamer V2 hyperparameters for DM control (which we use for all our environments). For the explorer, we use the default hyperparameters from the Dreamer V2 codebase [147]. We use action repeat of 2 following Dreamer. LEXA includes only one additional hyperparameter, the proportion of negative sampled goals for training the distance function. It is specified in Tab. L.5. The hyperparameters were chosen by manual tuning due to limited compute resources. The base hyperparameters are shared across all methods for fairness.

DIAYN baseline: We found that this baseline performs best when the reverse predictor is conditioned on the single image embedding e rather than latent state s. We use a skill space dimension of 16 with uniform prior and Gaussian reverse predictor with constant variance. For training, we produce the embedding using the embedding prediction network from Sec. 2.2.4. We observed that DIAYN can successfully achieve simple reaching goals using the skill obtained by running the reverse predictor on the goal image. However, it struggles with more complex tasks such as pushing, where it only matches the robot arm.

GCSL baseline: We found that this baseline performs best when the policy is conditioned on the single image embedding e rather than latent state s. This baseline is trained on the replay buffer images and only uses imagined rollouts to train an explorer policy. For training, we sample a random image from a trajectory and sample the goal image from the uniform distribution over the images later in the trajectory following [123]. We similarly observe that this baseline can perform simple reaching goals, but struggles with more complex goals.

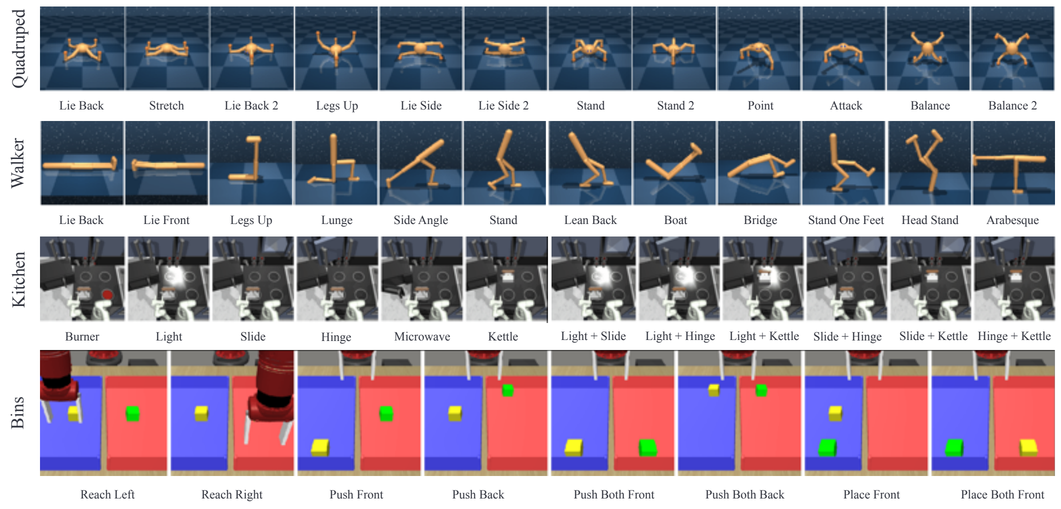
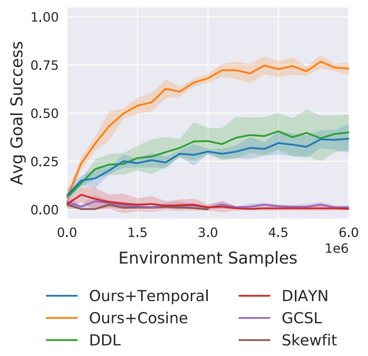
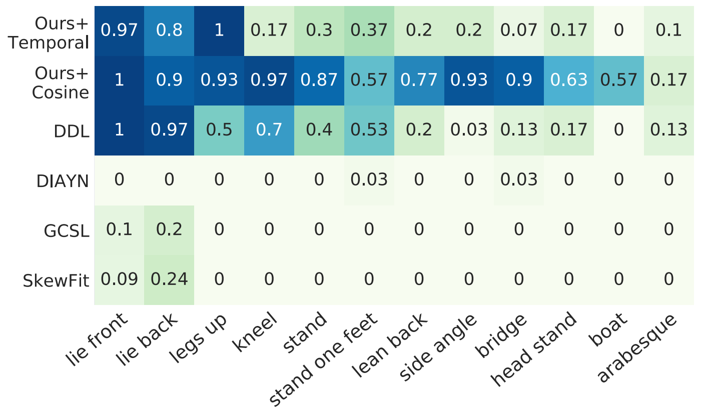


Figure L.1: All goals for the four environments that we consider. Our benchmark further includes an additional set of even harder goals, available in the repository.

Algorithm	**From Pixels**	**Zero-Shot**	**Exploration**	**Planning**
CTS [28], Curiosity [285], RND [44]	✓	✗	✓	✗
Plan2Explore [349]	✓	✗	✓	✓
HER [8]	✗	✓	✗	✗
Visual Foresight [97]	✓	✓	✗	✓
Actionable Models [55]	✓	✓	✗	✗
DIAYN [102]	✗	✓	✓	✗
Asymmetric Self-Play [275]	✗	✓	✓	✗
SkewFit [295]	✓	✓	✓	✗
Go-Explore [99]	✓	✓	✓	✗
LEXA (Ours)	✓	✓	✓	✓

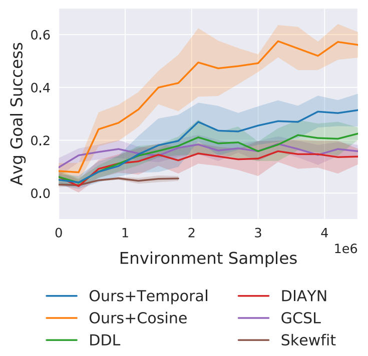
Table L.1: Conceptual comparison of unsupervised reinforcement learning methods. LEXA combines forward-looking exploration by planning with achieving downstream tasks zero-shot while learning purely from pixels without any privileged information.

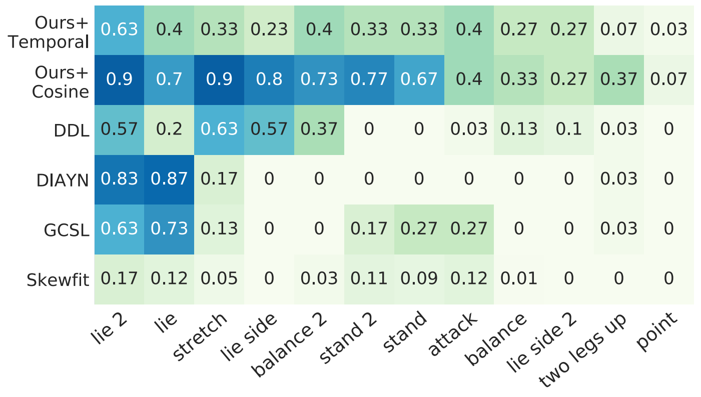




	lie front	lie back	legs up	kneel	stand	stand one feet	lean back	side angle	bridge	head stand	boat	arabesque
Ours+ Temporal	0.97	0.8	1	0.17	0.3	0.37	0.2	0.2	0.07	0.17	0	0.1
Ours+ Cosine	1	0.9	0.93	0.97	0.87	0.57	0.77	0.93	0.9	0.63	0.57	0.17
DDL	1	0.97	0.5	0.7	0.4	0.53	0.2	0.03	0.13	0.17	0	0.13
DIAYN	0	0	0	0	0	0.03	0	0	0.03	0	0	0
GCSL	0.1	0.2	0	0	0	0	0	0	0	0	0	0
SkewFit	0.09	0.24	0	0	0	0	0	0	0	0	0	0

Figure L.2: **RoboYoga Walker Benchmark.** Left: success rates averaged across all 12 tasks. Right: final performance on each specific task, ranging from light green (0) to dark blue (100%). We observe that the simple latent cosine distance function works well on this task, substantially outperforming other competing agents. In the heatmap, most agents can solve the easy tasks, but only LEXA makes progress on solving a majority of the tasks and achieves good performance.





	lie 2	lie	stretch	lie side	balance 2	stand 2	stand	attack	balance	lie side 2	two legs up	point
Ours+ Temporal	0.63	0.4	0.33	0.23	0.4	0.33	0.33	0.4	0.27	0.27	0.07	0.03
Ours+ Cosine	0.9	0.7	0.9	0.8	0.73	0.77	0.67	0.4	0.33	0.27	0.37	0.07
DDL	0.57	0.2	0.63	0.57	0.37	0	0	0.03	0.13	0.1	0.03	0
DIAYN	0.83	0.87	0.17	0	0	0	0	0	0	0	0.03	0
GCSL	0.63	0.73	0.13	0	0	0.17	0.27	0.27	0	0	0.03	0
Skewfit	0.17	0.12	0.05	0	0.03	0.11	0.09	0.12	0.01	0	0	0

Figure L.3: **RoboYoga Quadruped Benchmark.** Left: success rates averaged across all 12 tasks. Right: final performance on each specific task, ranging from light green (0) to dark blue (100%). We observe that the simple latent cosine distance function works well on this task, substantially outperforming other competing agents. In the heatmap, most agents can solve the easy tasks, but only LEXA makes progress on solving a majority of the tasks and achieves good performance.

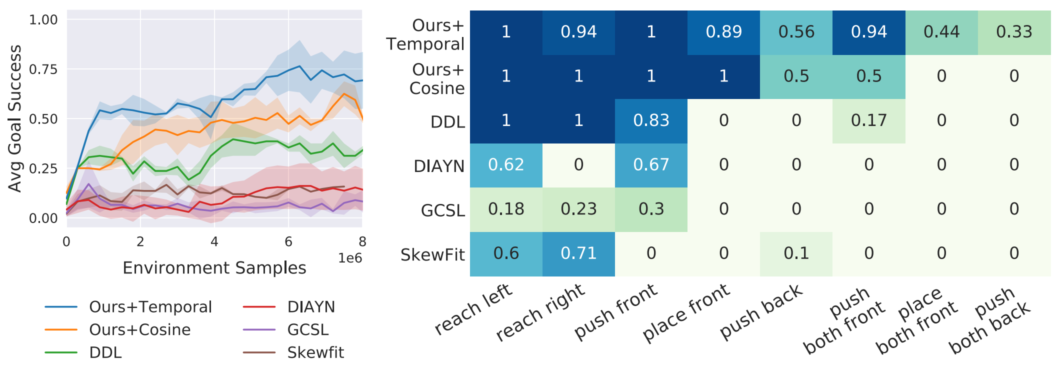


Figure L.4: **RoboBin Benchmark.** Left: success rates averaged across all 8 tasks. Right: final performance on each specific task. While cosine distance works on simple goals, temporal distance outperforms it on tasks requiring manipulating several blocks (last three columns), as this distance focuses on the part of the environment that's hardest to manipulate. Prior agents only solve the easiest reaching tasks, struggling either with exploration or learning the downstream policy.

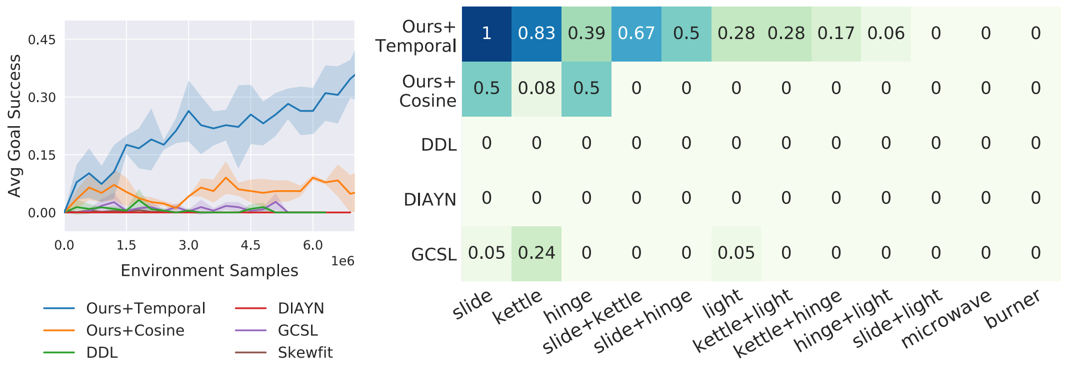


Figure L.5: **RoboKitchen Benchmark.** Left: success rates averaged across all 12 tasks. Right: final performance on each specific task. RoboKitchen is challenging both for exploration and downstream control, with most prior agents failing all tasks. In contrast, LEXA is able to learn both an effective explorer and achiever policy. Temporal distance helps LEXA focus on small parts such as the light switch, necessary to solve these tasks. LEXA makes progress on four out of six base tasks, and is even able to solve combined goal images requiring e.g. both moving the kettle and opening a cabinet.

Table L.5: Hyperparameters for LEXA over the Dreamer default hyperparameters.

Hyperparameter	Value	Considered values
Action repeat (all environments)	2	2
Proportion of negative samples	0.1	0, 0.1, 0.5, 1
Proportion of explorer:achiever data collected in real environment	1:1	1:1
Proportion of explorer:achiever training imagination rollouts	1:1	1:1

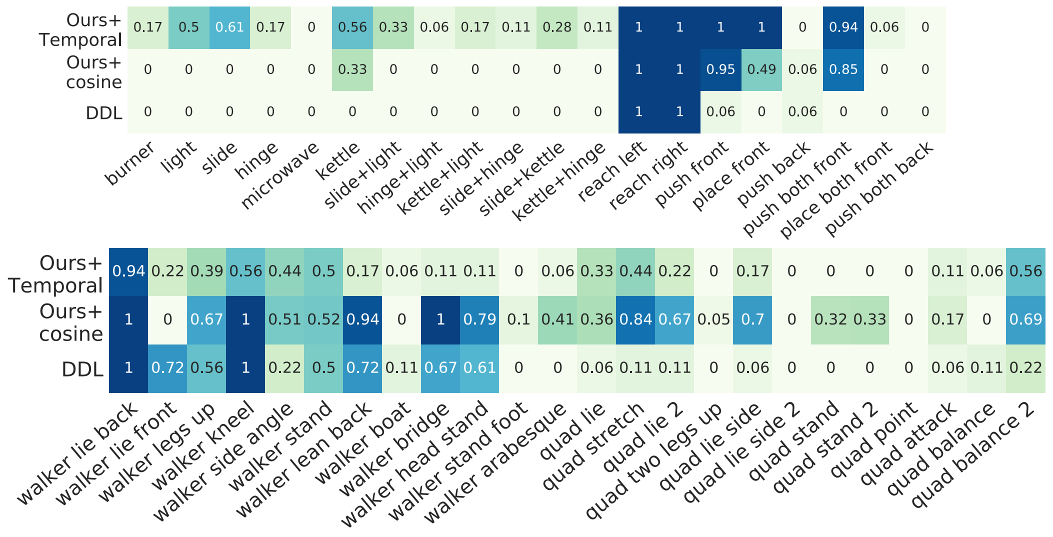


Figure L.6: **Single agent** trained across Kitchen, RoboBin, Walker, with final performance on each specific task. LEXA with temporal distance is able to make progress on tasks from all environments, while LEXA+cosine and DDL don't make progress on the kitchen tasks.

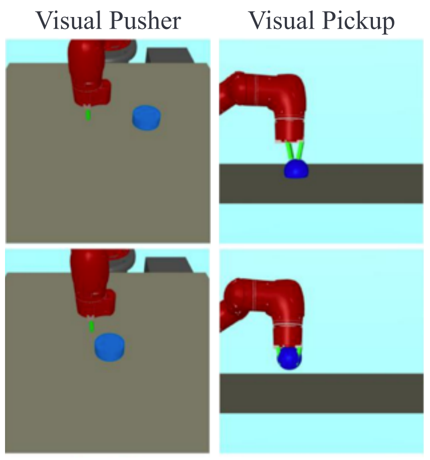


Table L.2: Results on SkewFit tasks [295].

Method	Visual Pusher	Visual Pickup
LEXA + temporal	**0.023**	**0.014**
Skew-Fit [295]	0.049	0.018
RIG [271]	0.077	0.037
RIG + Hazan et al.	0.059	0.039
RIG + HER [8]	0.075	0.035
DISCERN [414]	0.094	0.039
RIG + Goal GAN [109]	0.088	0.039
RIG + DISCERN-g	0.07	0.032
RIG + # Exploration	0.088	0.04
RIG + Rank-Based	0.067	0.035

Figure L.7: Final goal reaching error in meters on tasks from SkewFit [295]. Example observations are provided on the left. Baseline results are taken from [295]. LEXA significantly outperforms prior work on these tasks. Pushing and picking up blocks from visual observations is largely solved, so future work will likely focus on harder benchmarks such as the one proposed in our work.

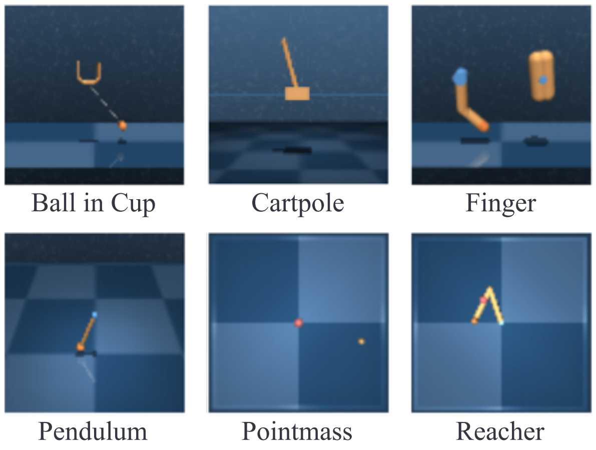


Table L.3: Results on DISCERN tasks* [414].

	LEXA	DISCERN* [414]
Ball in cup	**84%**	76.5%
Cartpole	**35.9%**	21.3%
Finger	**40.9%**	21.8%
Pendulum	**79.1%**	75.7%
Pointmass	**83.2%**	49.6%
Reacher	**100%**	87.1%

Figure L.8: Goal success rate on the tasks replicated from [414]. Example observations are provided on the left. *While the original tasks are not released, we followed the procedure for generating the goals described in [414]. Despite following the exact procedure, we were not able to obtain similar goals to the ones used in the original paper. Nevertheless, we show the goal completion percentage results obtained with our reproduced evaluation compared to DISCERN results from the original paper. We see that our agent solves many of the tasks in this benchmark and performs better on this comparison.

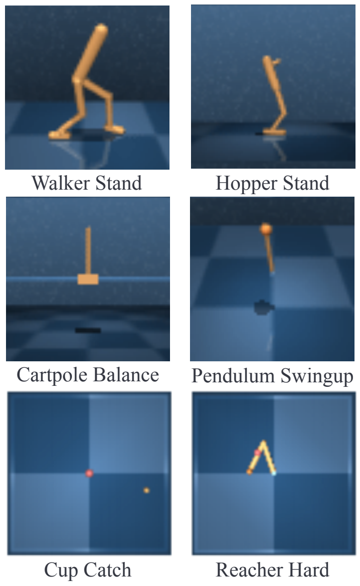


Table L.4: Results on zero-shot DeepMind control tasks from Plan2Explore [349].

Method	LEXA	P2E [349]	DrQv2 [441]
Zero-Shot	✓	✓*	✗
Walker Stand	**957**	331	968
Hopper Stand	**840**	**841**	957
Cartpole Balance	886	**950**	989
Cartpole Balance Sparse	**996**	860	983
Pendulum Swingup	**788**	**792**	837
Cup Catch	**969**	**962**	909
Reacher Hard	**937**	66	970

Figure L.9: Final return on DM control tasks [391]. Example goals achieved by LEXA are provided to the left. Baseline results taken from [349, 441]. Plan2Explore adapts to new tasks but it needs the reward function to be known at test time while LEXA does not require access to rewards. To compare on the same benchmark, we create goal images that correspond to the reward functions. Our agent even performs comparably to state of the art oracle agents (DrQ, DrQv2, Dreamer) that use true task rewards during training.

L.2 Affordances from Human Videos as a Versatile Representation for Robotics

L.2.1 Affordance Model Setup

Data Extraction: Our training setup involves learning from EpicKitchens-100 Videos [76]. This dataset contains many hours of videos of humans performing different kitchen tasks. We use each sub-action video (such as 'open door' or 'put cup on table') as training sequences. Consider a video (V) consisting of T frames, $V = \{I_1, ..., I_T\}$. Using 100 DOH annotations [356] (available alongside the dataset), we find all of the hand-object contact points and frames for each hand in the video. As mentioned in Section 3, let model output $f_{\text{hand}}(I_t) = \{h_t^l, h_t^r, o_t^l, o_t^r\}$, where o^l, o^r are the contact variables and h^l, h^r are the hand bounding boxes. We find the first contact timestep and select the active hand (left or right) as the hand side to consider for the whole trajectory. This is found by first binning o_t and looking for all types that have contact with 'Portable' or 'Fixed' objects. These are assigned 1, while all others are assigned 0. We smooth the set of contact variables using a Savitzky–Golay filter [337] using a threshold of 0.75 (with window size 7). This should eliminate any spurious detections. We use the skin segmentation approach from [236], to find the contact points, $\{c_i\}^N$, at the contact timestep around the active hand. We then fit a GMM with $k = 5$ to the set of contact points to determine $\mu_1, ..., \mu_5$. We found that learning without a covariance, Σ, was more stable thus we only aim to learn the μ_1. The input image becomes the first image before the contact where the hand is not visible. If the contact points or trajectory are not in the frame of this initial image (if the camera has moved), we then discard the trajectory. We use crops of size 150x150 (full image size is 456 x 256), which improves robustness at test time. We train on around 54K image-trajectory-contact point tuples. We include visualizations of the affordance model outputs on the VRB website.

Architecture: We use the ResNet18 encoder from [273] as g_ϕ, as our visual backbone. Our model has two heads, a trajectory head and a contact point head. We use the spatial features from the ResNet18 encoder (before the average pooling layer) as an input to three deconvolutional layers and two convolutional blocks with kernel sizes of 2 and 3 respectively, and channels: [256, 128, 64, 10, 5]. We use a spatial softmax to obtain $\hat{mu}_k$ for where $k = 1, ..., 5$. Our trajectory network is a transformer encoder with 6 self-attention layers with 8 heads each, and uses the output of the ResNet18 encoder (flattened), which has dimension 512. The output of the transformer encoder is used to predict a trajectory of length 5, using an MLP with two layers with hidden size 192.

Training: We train our model for 500 Epochs, using a learning rate of 0.0001 with cosine scheduling, and the ADAM [195] optimizer. We train on 4 GPUs (2080Ti) for about 18 hours.

L.2.2 Robotics Setup

Hardware setup: For all the tasks we assume the following structure for robot control for each trajectory. We first sample a rotation configuration for the gripper. The arm then moves to the contact point c, closes its gripper, and moves to the points in the post-contact trajectory τ. For the initial rotation of the Franka, joints 5 and 6 can take values in $[0, 30, 45]$ degrees, while joint4 is fixed to be 0 degrees. For the Hello-Robot, the roll of the end-effector is varied in the range of $[0, 45, 90]$ degrees. Once the orientation is chosen for the trajectory, we perform 3DOF end-effector control to move between points. Given two points a and b, we generate a sequence of waypoints between them to be reached using impedance control for the Franka. The Hello-Robot is axis aligned and has a telescoping arm, thus we did not need to build our own controller. We do not constrain the orientation to be exactly the same as what was selected in the beginning of the trajectory, since this might make reaching some points infeasible. For all tasks and methods we evaluate success rate by manual inspection of proximity to the goal image after robot execution (for imitation learning, goal reaching and affordance as an action space), and evaluate coincidental success for exploration using manual inspection of whether the objects noticeably move over the course of the robot's execution trajectory.

Affordance Model to Robot Actions: Reusing terminology from Section 3, the affordance model output is $f_\theta(I_t) = \hat{p}_c, \hat{\tau}$, where $\hat{p}_c = \sum_{k=0}^{K} \alpha_k \mathcal{N}(\hat{\mu_k}, \hat{\Sigma_k})$, and $\hat{\tau} = \{w_i\}^M$. We can convert this into a 3D set of waypoints using a hand-eye calibrated camera, and obtain a 3D grasp point from $\hat{p}_c$, and a set of 3D waypoints from $\hat{\tau}$.

Imitation from Offline Data Collection: We use our affordance model to collect data for different tasks, and then evaluate whether this data can be used to reach goal images using k-NN and Behavior cloning.

As mentioned in Sec 3.3.1, given an image I_t, the affordance model produces $(c, \tau) = f_\theta(I)$. In addition to storing I_t, c and τ, we also store the sequence of image observations (queried at a fixed frequency) seen by the robot when executing this trajectory $O_{1:k}$, where k is the total number of images in the trajectory. k varies across different trajectories (since it depends on c and τ). These intermediate images O_i enable us to determine how close a trajectory is to the given goal image. For each trajectory, the distance to goal image I_g is given by $\min_i ||\psi(I_g) - \psi(O_i)||_2^2$, where ψ is the R3M embedding space. We then use this distance to produce a set of K trajectories with smallest distances to the goal I_g. For k-NN, we simply run (c, τ) from each of these filtered trajectories. For Behavior cloning, we first train a policy that predicts (c, τ) given image I using

	Cabinet	**Knife**	**Veg**	**Shelf**	**Pot**	**Door**	**Lid**	**Drawer**
N_0	150	100	50	50	50	50	30	40
N_s	50	50	30	30	30	50	30	40

Table L.6: Number of trajectories collected for various tasks, for Initial Data Collection (N_0) and for each subsequent fitting iteration for either goal reaching or exploration (N_s)

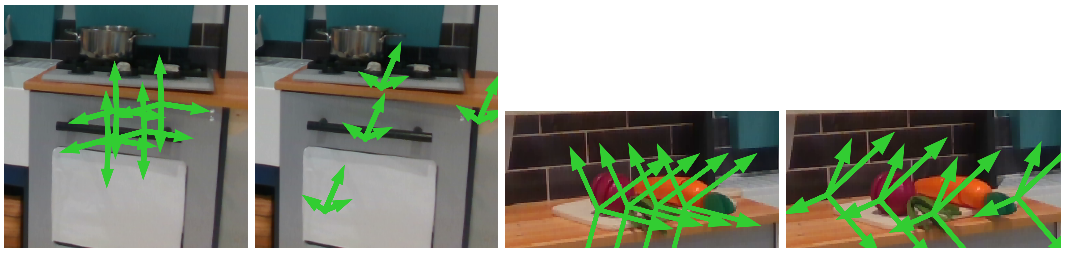

(a) VRB - Cabinet (b) HAP - Cabinet (c) VRB - Veggies (d) HAP - Veggies

Figure L.10: Visualization for Affordance as an Action Space for VRB and HAP [128], on the Cabinet and Veggies Tasks

this set of trajectories, and then run the policy π on the robot. We summarize this is Algorithm 10. We fix the number of top trajectories K to be 10 for k-NN and 20 for behavior cloning. The number of trajectories for initial data collection used for each task is listed in L.6.

For k-NN, the success is averaged across all K runs on the robot. For behavior-cloning, we parameterize the policy π using a CVAE, where the image is the context, the encoder and decoder are 2 layer MLPs with 64 hidden units and the latent dimension is 4. During inference, we sample from the CVAE given the current image as context, and report success averaged across 10 runs. The quality of data collected by the robot using VRB which is used for imitation can be in seen on the VRB website.

Object	VRB	Hotspots
VR Controller	**0.27**	0.13
Chain	**0.33**	0.20
Hat	0.07	**0.20**
Tape	**0.13**	0.00
Cube	0.00	0.00
Sanitizer	**0.27**	0.20
Stapler	**0.53**	0.20
Shoe	**0.33**	0.13
Mouse	**0.27**	0.00
Hair-Clip	**0.47**	0.20

Table L.7: VRB for grasping held-out objects

Although many of our household object categories might be present in the videos of Epic-Kitchens [76], specific *instances* of objects do not appear in training, thus every object our approach is evaluated on is new. To test generalization to "rare" (held-out) objects and evaluate the grasping success using VRB's affordances, see Table L.7. VRB consistently outperforms our most competitive baseline, Hotspots [267].

Exploration & Goal Reaching: We apply our affordance model in the paradigms of exploration as well as goal reaching, where the robot uses the collected data to improve its behavior. As described in Section 3.3, we use a *environment change* visual model to obtain intrinsic reward for exploration, while for goal-reaching we use *distance to the goal* in a feature space like the R3M embedding space. For exploration,

Algorithm 10 Imitation from Offline Data Collection

```
Require: Dataset of trajectories $\{(I_t, O_{1:k}, c, \tau)\}$
Require: Number of top trajectories K
Require: Goal Image $I_g$
Require: R3M embedding space $\psi$
 1: For each trajectory $\mathcal{T}$, compute
 2: $d_{\mathcal{T}} = \min_i ||\psi(I_g) - \psi(O_i)||_2^2$
 3: Rank trajectories in ascending order of $d_{\mathcal{T}}$. Create set $\mathcal{K} = \{(c, \tau)\}$ of the top K
    ranked trajectories.
 4: if $k$-NN then
 5:     Execute $\mathcal{K}$ on the robot.
 6: else
 7:     Assert behavior cloning
 8:     Train a policy $\pi(c, \tau | I)$ using $\mathcal{K}$.
 9:     Execute $c, \tau \sim \pi(.|I)$ on the robot.
10: end if
```

we want to *maximize* the change between the first and last images of the trajectory, since greater perturbation of objects can lead to the discovery of useful manipulation skills. For goal-reaching, we *minimize* the distance between the trajectory and the goal image, since this achieves the desired object state. In each case (exploration and goal-reaching), we rank the trajectories in the dataset using the appropriate metric, and then fit $(\hat{c}, \hat{\tau})$ to the $\{(c, \tau)\}$ values of the top ranked trajectories. For subsequent data collection iterations, we use the affordance model f_θ with some probability p, but otherwise use $(\hat{c}, \hat{\tau})$ for execution on the robot. The newly collected data is then aggregated with the dataset, and the entire process repeated. We present this procedure in Algorithm 11. The number of initial trajectories N_0 and trajectories for subsequent iterations N_s for different tasks are listed in L.6. For all experiments, we set p = 0.35, K = 10, J = 2. We include videos on the VRB website. which show that as our system sees more data, its performance improves for both exploration and goal-reaching.

Intrinsic Reward Model: We train a visual model which given a pair of images (I_i, I_j), produces a binary image that captures how *objects* move, and is not affected by changes in the robot arm or body position. Specifically, this model comprises the following -

$$\begin{aligned}\phi(I_i, I_j) = g(&||m(I_i) - m(I_j)||_2, \\ &||\Psi(m(I_i)) - \Psi(m(I_j))||_2)\end{aligned} \tag{L.1}$$

Here m is a masking network which removes the robot from the image. We train this using around 100-200 hand-annotations of the robot in various scenes, and use this data to fine-tune a pretrained segmentation model Ψ [158]. We evaluate the l2-

Algorithm 11 Exploration / Goal Reaching

```
Require: Number of iterations J
Require: Number of top trajectories K
Require: Number of initial trajectories $N_0$,
 1: and for subsequent fitting iterations $N_s$
Require: Affordance model $f_\theta$
Require: Tradeoff probability $p$
Require: Visual change model $\Phi$ (only for exploration)
Require: R3M embedding $\psi$ (only for goal reaching)
Require: Goal Image $I_g$ (only for goal reaching)
 2: initialize: World model $\mathcal{M}$, Replay buffer $\mathcal{D}$,
 3: Execute $(c, \tau) = f_\theta(I)$ on the robot for $N_0$ iterations to collect initial dataset $\mathcal{D} = \{(I, O_{1:k}, c, \tau)\}$
 4: for iteration 1:J do
 5:     For each trajectory $\mathcal{T}_{0:k}$, compute
 6:
 7:     if exploring then
 8:         compute $\text{EC}_\mathcal{T} = ||\phi(O_1) - \phi(O_k)||_2$
 9:         Rank trajectories in descending order of $\text{EC}_\mathcal{T}$
10:     else
11:         Assert goal reaching
12:         compute $\text{d}_\mathcal{T} = \min_i ||\psi(I_g) - \psi(O_i)||_2$
13:         Rank trajectories in ascending order of $d_\mathcal{T}$
14:     end if
15:     Create set $\mathcal{K} = \{(c, \tau)\}$ of top K ranked trajectories.
16:     Compute $\hat{c}, \hat{\tau}$ = mean($\mathcal{K}$)
17:     For $N_s$ iterations, set $(c, \tau) = f_\theta(I)$ with probability $p$, otherwise set $(c, \tau) = (\hat{c}, \hat{\tau})$.
18:     Execute $(c, \tau)$ on the robot and append data to $\mathcal{D}$
19: end for
```

losses above only on **non-masked** pixels. Further, we also take into account distance in the feature space of the segmentation model to reduce sensitivity to spurious visual artifacts. The function g applies heurestics including gaussian blurring to reduce effects of shadows, and a threshold for the change at each pixel, to limit false positives.

Affordance as an Action Space: For this learning setup, we parameterize the action space for the robot with the output distribution of our affordance model. We first query the model a large number of times, and then fit Gaussian Mixture Models (GMMs) separately to the c and τ predictions, with N_c and N_τ centers respectively. We then define a discrete action space of dimension N_c*N_τ, where each action maps

Algorithm 12 Affordance as Action Space

```
Require: Affordance Model $f_\theta$
Require: Number of initial queries $q$
Require: Number of clusters for $c$, $N_c$ and for $\tau$, $N_\tau$
Require: Goal Image $I_g$
Require: RL algorithm with discrete action-space $RLA$
Require: R3M embedding space $\psi$
 1: Query $f_\theta$ on the image of the scene q times
 2: to obtain a dataset $\{(c, \tau\}$
 3: Fit a GMM $G_c$ with $N_c$ centers to $\{c\}$, and
 4: a GMM $G_\tau$ and $N_\tau$ centers to $\{\tau\}$
 5: Create mapping $\mathcal{M}$ from $\mathcal{A}$ = $[1..N_c$*$N_\tau]$ to values in the cross-product space of
    the centers of $G_c$ and $G_\tau$
 6: Initialize Dataset $\mathcal{D}$ = {}, and $RLA$ with discrete action space $\mathcal{A}$ and random
    policy $\pi$.
 7: Run Sampling and Training asynchronously
 8: while Sampling do
 9:     Run $\pi$ on the image to get $a_d$.
10:     $(c, \tau) = \mathcal{M}(a_d)$, execute on the robot and collect initial and final images $I_0$
    and $I_T$
11:     Compute reward $r = ||\psi(I_T) - \psi(I_g)||_2$.
12:     Store $(\psi(I_0), a_d, \psi(I_T), r)$ in $\mathcal{D}$
13: end while
14: while Training do
15:     Sample data $\sim \mathcal{D}$, pass to $RLA$ for
16: training and updating $\pi$.
17: end while
```

to a value in the cross-product space of the centers of the two GMMs. We can now use discrete action-space RL algorithms. We asynchronously sample from the discrete action-space policy, and train it using the RL algorithm. This procedure is described in Algorithm 12. We note that it is important to reset the environment so that images the policy sees are close to the initial image for which the action space was defined. Across experiments we set N_c = N_τ = 4, q = 2000. For the RL algorithm RLA we use the DQN [258] implementation from the d3rlpy [388] library. We include a visualization of the action space by plotting the (c, τ) values in the cross-product space of the centers of the two GMMs, for VRB and HAP [128] in Figure L.10. For VRB, a larger number of the discretized actions are likely to interact with objects.

L.2.3 Baselines and Ablations

Baselines: The baselines we compare to include the approaches from from Liu et al. [236] (HOI), Goyal et al. [128] (HAP) and Natarajan et al., (Hotspots) [267]. In each of these baselines, we used the provided pretrained model. Specficially, for Hotspots [267], we employ the model trained on EpicKitchens [76], as this is what our approach is also trained on. Similarly, for HAP [128] we use the trained model on EpicKitchens also. HOI predicts both a contact point and trajectory, which we execute at test time. The other two approaches predict likely contact regions, from which we sample, as well as a random post contact trajectory.

Visual Representation Analysis (Fine-tuning): For the visual representation fine-tuning experiments we performed in Section 4.5, we use the Imitation Learning Evaluation Framework from R3M [273], which aims to evaluate the effectiveness of frozen visual representations for performing behavior cloning for robotic control tasks. Following their procedure, we evaluate on three simulated tasks from the Franka Kitchen environment: (1) microwave, (2) slide-door, and (3) door-open. We train the policy using left camera images from their publicly available demonstration dataset, which is collected by an expert state-based reinforcement learning agent and then rendered as image observations. For behavior cloning with the R3M encoder, we freeze the pretrained R3M encoder (which uses a ResNet50 base architecture) and fine-tune a policy on top of it. For behavior cloning with the VRB encoder, we use an R3M model which was fine-tuned for 400 steps with affordance model training as in Section 3.2. fine-tuning was performed separately from behavior cloning, and during policy learning our representations are also frozen before being used as input for the downstream policy. For both R3M and VRB, we concatenate the visual embedding and pro-prioceptive data for input to the downstream policy, and then use a BatchNorm layer followed by a 2-layer MLP to output an action, and use a learning rate of 0.001 and a batch size of 32 for 2000 steps.

Visual Representation Analysis (Feature space distance): For the feature space distance experiments, we compare an R3M model with a VRB model. Both use a ResNet50 base architecture, and the VRB model is obtained by fine-tuning an R3M model for 100 steps using affordance model training as in Section 3.2. The distances in Figure 8 are computed as the (squared) L2 distances between the features produced by each model for the goal image and current image.

Method	Light	Microwave	Kettle
Random	0.20	0.15	0.20
HAP	0.30	0.20	0.45
HOI	0.60	0.45	0.40
Hotspots	0.35	0.35	0.25
VRB	**0.75**	**0.60**	**0.55**

Table L.8: VRB on simulation benchmarks.

L.2.4 Simulation

We also provide a simulation environment benchmark to test our affordances. This is modeled after the Franka-Kitchen environment from the D4RL [114]. In this benchmark, the robot observes images and predicts 3D positions to manipulate, in the exact same way as we deploy the robot in the real world. There are three different tasks: turning the light on, opening the microwave and lifting the kettle. These are standard tasks in the D4RL benchmark [114]. We run Paradigm 1 (offline data collection) and provide the success rates for VRB and baselines in Table L.8.

L.2.5 Codebases

We use the following codebases:

- epic-kitchens/epic-kitchens-100-hand-object-bboxes for extracting detections from 100 DOH [356] for EpicKitchens [76].
- stevenlsw/hoi-forecast for Skin segmentation code and HOI baseline [236].
- uiuc-robovision/hands-as-probes for HAP baseline [128].
- Tushar-N/interaction-hotspots for Hotspots baseline [267].
- facebookresearch/r3m for R3M visual features [273].
- wkentaro/labelme for getting masks for robot and
- Torchvision tutorial for a Mask-RCNN [158] implementation.
- takuseno/d3rlpy [388] for DQN [258] implementation.
- facebookresearch/polymetis [229] as the base for the controller for the Franka Arm.

L.3 Structured World Models from Human Videos

L.3.1 Robot Setup

We use two robots: Franka Emika and Stretch RE1 from Hello-Robot. Both robots are controlled in end-effector space as well as a rotation (roll for the Stretch, and roll + pitch for the Franka). The Franka roll and pitch, as well as the Stretch roll are sampled from $[0, -\frac{\pi}{4}, -\frac{\pi}{2}]$ (randomly at first). The robots run open loop trajectories. The camera observations are coming from D415 Intel Realsense RGBD cameras. We use a low-level impedance controller for the Franka to reach the desired high level actions.

L.3.2 Tasks and Environments

Our setup consists of six tasks, three (veggies, knife and cabinet) of which are in a Play Kitchen from Ebert et al. [98], and we have three in the wild tasks that involve opening the dishwasher, lifting the garbage can handle or pulling out a drawer. These are everyday tasks that we found. Videos of each task can be seen at `https://human-world-model.github.io`.

L.3.3 Data Collection

We perform data collection by executing the affordance model $\mathcal{G}_\psi$ at first, and then set mode $m_t = 1$, and use Δy_t as Cartesian end-effector deltas. These are sampled from $\mathcal{N}(0, 0.05)$. We sample from $\mathcal{G}_\psi$ in the following manner: we obtain the 2D pixel from the model, and find the depth at that point. For the grasp part of the affordance, we simply pass this depth to our controller (which has hand-eye calibration). For post grasp trajectory, we sample a depth with d as the center, with a bias towards moving away from the surface (as we usually have a wall right behind the object, or the depth camera). For each of the baselines, we use the underlying action space to sample actions, and append Δy_t to the end. Our trajectory, during the robot sampling stage, consists of 3-4 actions with $m_t = 0$, and 6-10 actions with $m_t = 1$. The overall data collection process takes about 25 to 45 minutes depending on how long resets takes.

L.3.4 Human Videos

Our human video dataset is obtained from Epic-Kitchens [76]. We take semantically pre-annotated action clips, and apply the 100 Days of Hands (100 DoH) [356] hand-object model to get annotations for when and where the contact happened, and how the hand moved post contact, all in normalized (0, 1) pixel space. To obtain the contact points, we use a similar pipeline to Liu et al. [236], where we find the intersection of the hand bounding box and the interacted object's bounding box, and look for skin outline in that region. We use a skin segmentation (similarly to

Liu et al. [236]) to get the external grasp points. We obtain about 55K such clips to train on. Each sequence is of length 4, with $m_t = 0$ for all t. For the rotation and the depth values, we randomly sample these values during training, from one of the feasible rotations, or within 50cm of the environment surface respectively. We train a ResNet18 based encoder-decoder architecture for our grasp point prediction. We perform a spatial softmax on the decoder deconvolutional output to obtain the grasp keypoints. The post-grasp trajectory head is a Transformer [402] with 6 self-attention layers that have 8 heads, inspired from Liu et al. [236].

L.3.5 World Model

Our world model architecture is the same as that of Hafner et al. [147], excluding the visual encoders or decoders. We do not tune any of the world model hyper-parameters, and use the default Dreamer[147] settings. We use the NVAE [400, 16] encoder and decoder used in FitVid [16] to better handle high dimensional image prediction. We use only one cell per block instead of two, due to GPU memory restrictions and to train with larger batch sizes. We do not have any residual connections between the encoder and the decoders, to force the latent of the world model, z, to be an information bottleneck. The dimension of z is 650 (the deterministic component of the RSSM[147] is size 600, and the stochastic component is size 50). The model is trained in Tensorflow, and each image is of the size 128x128x3. In the experiments that use reward prediction, we regress q_r (the reward decoder) to the distance to goal in the space of R3M [273] features (the ResNet18 [157] version) of the weights. The reward predictor network consists of a 2 layer MLP with 400 hidden units which takes the world model feature z as input.

L.3.6 Baselines

Every baseline that uses a world model uses the same code as SWIM, with either a different pre-training setup or different action space.

- `MBRL-Affordance`: This is the same exact setup as SWIM in terms of the world model and the execution of the affordance model, but we do not use any pre-trained weights when training on robot data.
- `MBRL-Pix`: The action type is the same as `MBRL-Affordance`, but the pixel locations are chosen at random, and not from the human-centric affordance model. The actions are sampled uniformly in the 2D crop around the object.
- `MBRL:` Here all of the actions are with $m_t = 1$.
- `BC-Affordance:` This is a filtered-behavior cloning [289, 291] strategy. We rank trajectories based on distance in R3M [273] space to goal. We fit a Gaussian Mixture Model with 2 centers to the top actions, and sample from those, at execution time.

- `BC-Pix`: We fit a GMM top trajectories just like `BC-Affordance`. The sampling space is uniform in the crop around the object.

L.3.7 Training, Finetuning and Deployment

For training the world model, $\mathcal{W}_\phi$, in each iteration we train on 100 batches of data, where each batch consists of an entire trajectory sequence. These sequences are of length 2, 3 and 10 for the human video, Hello robot and Franka robot settings respectively. We first train a model on the human data for about 6000 such iterations with a batch size of 80, which takes about 96 hours on a single RTX 3090 GPU (using 24GB of VRAM). We then fine-tune this model for 300 epochs on robot data for the joint model, and 200 iteration for the single-task models using a batch size of 24, on a RTX 3090, which takes about 3-4 hours of training. The batch size for robot data is smaller because the model needs to deal with longer sequences consisting of hybrid actions (both the affordance actions and cartesian end-effector actions). For the continual learning experiments we subsequently train on the aggregated datasets for an additional 50 iterations. When deploying the model to perform a task, we use CEM for planning at the beginning of the trajectory, and then execute the optimized action sequence in an open-loop manner. We use 3 iterations of CEM, and 2000 action proposals. Further, in all our experiments, we fix $\mathcal{M}$ = 1400 and $\mathcal{N}$ = 600 ($\mathcal{M}$ and $\mathcal{N}$ are defined in Alg. 2), for fixing the ratio of biasing the proposals sent to the model for planning.

L.3.8 Evaluation

We evaluate our world model by executing the trajectory it outputs in the real world using open-loop control. We use goal images that indicate objects are manipulated in specific ways, for example an open cabinet, vegetables picked up and in the air, the knife should be lifted up, the drawer pulled out, the garbage can and dishwasher opened. We evaluate for each method/ablation 25 times, presenting the average.

L.3.9 Codebases

- `https://github.com/danijar/dreamerv2` [147] for the world model code
- `https://github.com/ddshan/hand_object_detector`-100DoH model [356]
- `https://github.com/epic-kitchens` for Epic-Kitchens [76] processing
- `https://github.com/facebookresearch/r3m` for R3M [273] model
- `https://github.com/facebookresearch/fairo/tree/main/polymetis` [229] for the end-effector control code for the Franka
- `https://github.com/orgs/hello-robot/repositories` for Stretch RE-1

L.4 Continuously Improving Mobile Manipulation with Autonomous Real-World RL

L.4.1 Policy Training

For our experiments we run DrQ implemented in the official RLPD codebase open-sourced by Ball et al. [23]. Since we run image-based real robot experiments, we use learning algorithm hyperparameters (including for the image encoders) from Stachowicz et al. [378], which deployed RLPD for race car driving. The observations are first encoded into a latent space (separately for the actor and critic), and the processed latent is used by the critic ensemble or the actor. Details of the architecture for each of these, in addition to hyperparameters for training is provided in Table L.9. We use both image and vector observations for learning. Each of these is processed by an image encoder or a 1-layer dense encoding for vector observations, and the corresponding latents are all concatenated together and then used as input for the actor or critic. Note that we use separate encoders for the critic and the critic. We use the architecture from Stachowicz et al. [378] for encoding each image source, without using any pre-trained embeddings, the network is retrained from scratch for each new experiment. There are 4 RGB image sources. The network encoders are provided with the last 3 frames for each image source, except for the goal image, since this remains fixed for the episode. The image sources are -

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

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

- Body pose - We compute the (x,y,θ) position of the robot body in the SE(2) plane relative to the calibrated playpen frame (calibration details in section L.4.6). The input to the network is 4 dimensional, consisting of $(x, y, \cos(\theta), \sin(\theta))$. We use $\sin, \cos$ transforms for the angle to avoid discontinuities in input, since $-\pi$ and π represent the same orientation.
- Hand pose - This contains the 6-dof end effector orientation of the hand relative to the base position.

There are certain learning parameters that are tuned separately for each environment, which we list in Table L.10. This was mainly to balance the exploration-exploitation

Table L.9: Hyperparameters used in the experiments

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

trade-off for learning new behavior, and pertain to the weight placed on entropy maximization in DrQ (temperature and target entropy), or to handle sparse rewards (number of min Q functions). We use a maximum episode length of 16 for the chair and sweeping tasks, and 8 for the dustpan task, since it has sparse reward.

Table L.10: Environment-tuned Hyperparameters

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

L.4.2 Detection-Segmentation

For each task, there is an object of interest, the state of which is used to compute the reward. We specify the object using a text prompt, which is used by the detection model to obtain a bounding box. This is then used to condition the Segment Anything [197] model to obtain a 2D object mask, as shown in Fig.L.11. For text-based detection we use either Grounding-Dino [237] or Detic [462].

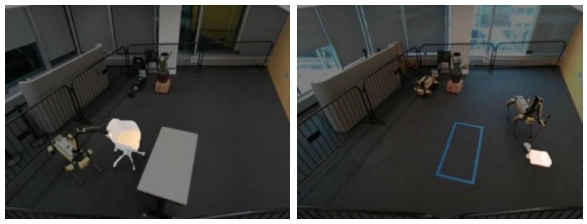


Figure L.11: **Grounded SAM/Detic Visualization**: Visualization of the object masks obtained from Segment Anything for chair moving(left) and sweeping (right).

For Grounding-Dino, we append the task-specific prompt to the list of class names in COCO [228] (to avoid cases of false positive detection), and we use Detic with `objects365` vocabulary class names. The task-specific text prompts we use are 'chair' for the chair tasks, 'red broom' for the dustpan standup task, and 'box.bag.poster.signboard.envelope.tag.clipboard.street_sign' for the sweeping task. The object of interest in the sweeping task is a paper bag being swept and we use many different possible matching text descriptions since it is detected as different classes due to its deformable nature. We list the detection model and the confidence threshold for a detection to be accepted for each task in Table L.11.

Table L.11: Detection Settings

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

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

L.4.3 Reward Function

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

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

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

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

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

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

L.4.4 Success Criteria

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

- Chair tasks - If the max reward obtained in the epsiode is above 1. This implies the chair is very close to its target.

- Dustpan Standup - If the episode ends with a reward of 10 (indicating the dustpan is standing up).

- Sweeping - If the episode ends with a reward of 10 (indicating the paper bag is swept into the goal region).

For the quantitative results in Fig 4 and 5, we compute the success/mean reward for each trajectory from a single run on the real robot system. These trajectories are then binned, with 20 trajectories per bin. We then compute the mean and standard error within each bin, and then plot against the corresponding number of real world interaction samples.

L.4.5 Priors

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

Algorithm 13 Prior generation for Dustpan Standup

```
1:  Initialize Prior data buffer $\mathcal{D}$
2:  Initialize Uniform noise distribution $\mathcal{U}$ with limits :
3:  $(-0.1, -0.1, -1) \rightarrow (0.1, 0.1, 1)$
4:  for $N = 1$ to Number of episodes do
5:      Initialize action list $\mathcal{A} = []$
6:      Set yaw hand rotation $\omega$ to +/-0.5
7:      for $t = 1$ to episode len do
8:          Set vertical hand action $z$ to +/-0.2
9:          Add $(z, \omega, 0) + (n \sim \mathcal{U})$ to $\mathcal{A}$
10:     end for
11:     Add $(-0.2, \omega, 0) + (n \sim \mathcal{U})$ to $\mathcal{A}$
12:     Execute $\mathcal{A}$ on the robot, record observations, add to $\mathcal{D}$
13: end for
14: return Prior data buffer $\mathcal{D}$
```

For sweeping, the prior is simply to stay within 0.5m of the last detected location of the paper bag. For dustpan standup we use a simple procedural function to generate trajectories to create a prior dataset, which we detail in Algorithm 13

L.4.6 Map Calibration

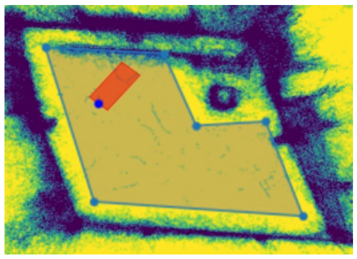


Figure L.12: Collision map of the playpen used for safety and navigation. The table is added to this map when included in experiments.

We use the GraphNav functionality provided in the SpotSDK by Boston Dynamics for Spot robots for generating a map of the playpen. This involves walking the robot around with some fiducials (we use 5) in the arena. This needs to be performed only once, and is used to obtain a reference frame to localize the robot, which is useful to record body pose information and also to implement safety checks to make sure the robot is not executing actions that collide with the playpen railings. While Spot has inbuilt collision avoidance we implement an additional safety layer using the map to clip unsafe actions that would move the robot too close to the playpen railings. 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. L.12. The red region denotes the estimate of the robot's position in the x-y plane, with the blue marking denoting its heading.

L.4.7 System Overview

We use a workstation with a single A5000 GPU to run RLPD online, which requires about 20GB GPU memory, mostly owing to all the image inputs that need to be processed. The detection and segmentation models are run on cloud compute on a single A100 GPU. The fixed third person camera images from the Realsense are streamed to a local laptop. Communication between the laptop, workstation and cloud server is facilitated via GRPC servers, and the main program script is run on the workstation, which also controls the robot. Commands are issued to the robot over wifi using the SpotSDK provided by Boston Dynamics.

L.5 Adaptive Mobile Manipulation for Articulated Objects in the Open World

L.5.1 Grasp Primitive

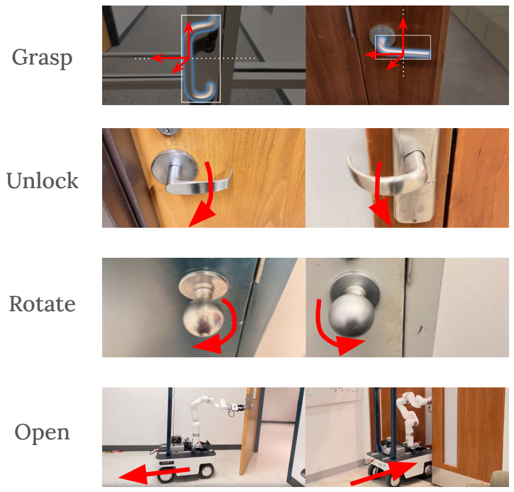


Figure L.13: **Primitives**: We show visualization of the primitives we used in our adaptive learning framework.

At the testing time, the robot is initialized randomly in front of the objects. Given the RGBD image of the scene obtained from the realsense camera, we use off-the-shelf visual models [462, 197] to obtain the mask of the door frame using just text prompts. Furthermore, since the door is a flat plane, we can estimate the surface normals of the door using the corresponding mask and the depth image. This is used to move the base close to the door and align it perpendicularly.
We further obtain a naive grasp pose of the handles from the detection and segmentation models [462, 197]. As shown in fig:door- L.13, given a text prompt of ”handle”, the open-vocabulary detection model [462] returns a 2D bounding box of the handle. As shown in the left image of the grasp examples in fig:door- L.13, if the width of the 2D bounding box is smaller than the length of the bounding box, we determine it is a vertical handle. Otherwise, it is a horizontal handle. The grasp orientation is determined by the surface normal of the door frame, the vertical-horizontal type of the handle, and the direction of gravity. We draw a dotted middle line to find the center point of the segmentation mask of the handle, and then it is projected into 3d coordinates using camera calibration and depth. However, passive detection and segmentation models are insufficient to predict a robust grasp pose for all handle types. For the grasp primitive, we introduce a 3-dimension continuous low-level parameter ranging from -1 to 1 as the grasp primitive input, which is then rescaled to the grasp offset ranging from $-d$ to d. This is beneficial since our adjusted grasp can be adapted to diverse handles via online adaptation.
An important aspect of the design of our gripper is that it enables compliant grasps. This is due to its hooked shape, as shown in Fig L.14 which allows for dynamically adjustable grasps as opposed to rigid immovable fixed grasps. Consider for example, the execution of the open primitive. The contact point between the door handle and the hooked-shaped finger shifts as the base steps back. This passive compliance allows the robot to manipulate the door with a force that *changes direction*, despite applying a simple base velocity command along the X axis of the base frame. It is

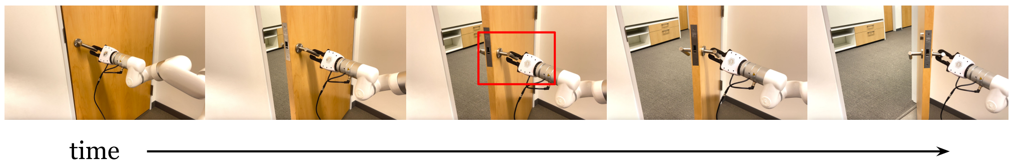


Figure L.14: **Compliant Grasp:** We design a hooker-shaped finger to allow a compliant grasp. As shown in the annotated red box, the contact point between the door handle and the hooker-shaped finger shifts as the robot base steps back, enabling passive compliance.

worth noting that different objects need different learned low-level parameters to determine how much to move, while avoiding unsafe primitive execution.

L.5.2 Constrained mobile manipulation primitive

We introduce three primitives, including unlock, rotate, and open. Each primitive is a functional API that takes a low-level parameter as the input to instantiate constrained mobile-manipulation action executions. We define two coordinate frames in the mobile manipulation system. We have a base frame, and an arm end-effector frame. The end-effector frame is defined relative to (i.e. with respect to) the base frame. With a 3-DOF motion for the base (in the SE(2) plane), and a 6-DOF arm (with respect to the base frame), we have a 9-dimensional vector -

$$(v_x, v_y, v_z, v_{\text{yaw}}, v_{\text{pitch}}, v_{\text{roll}}, V_{\text{x}}, V_{\text{y}}, V_\omega)$$

The first 6 dimensions correspond to velocity control for the arm end-effector, and the last three are the velocity control for the base. The primitives we use impose contraints on this space as follows -

$$\begin{aligned} \text{Unlock}: &\quad (0, 0, v_z, v_{\text{yaw}}, 0, 0, 0, 0, 0) \\ \text{Rotate}: &\quad (0, 0, 0, v_{\text{yaw}}, 0, 0, 0, 0, 0) \\ \text{Open}: &\quad (0, 0, 0, 0, 0, 0, V_{\text{x}}, 0, 0) \end{aligned}$$

The velocities of these primitives are a fixed value, and the low-level parameters are continuous one-dimension values ranging from -1 to 1, which is then rescaled to the primitive execution time ranging from $-T$ to T.

L.5.3 Extensive Evaluation with Enhanced Primitive

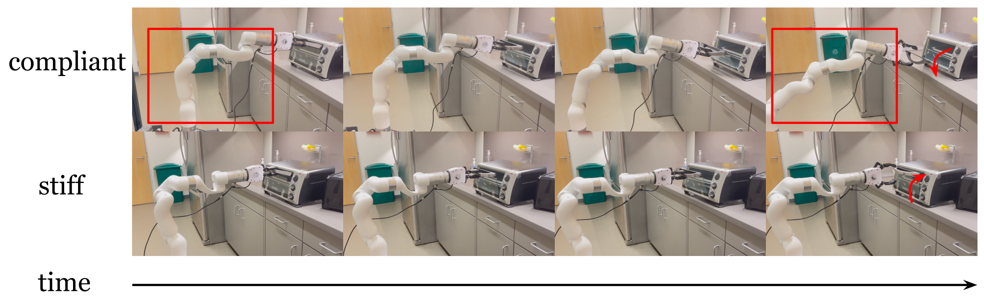


Figure L.15: **Enhanced Primitives with Force Torque Sensors:** The original open primitive struggles with articulating oven objects because it only controls the velocity of the base in SE(2), and a stiff arm cannot exert sufficient force with a changing direction to operate the oven. To enhance the open primitive, we install a force torque sensor on the robot end-effector to provide passive compliance. As shown in the annotated red box, the passive compliance allows the arm to align with the force direction of the articulation, while the robot base moves backward.

For better compliance involving control of the arm, we also experiment with using force torque sensors. In this section, we evaluate the effectiveness of our predefined primitives on a broader set of objects, specifically ovens, where the joint connecting the base part to the frame part remains perpendicular to the direction of gravity, as shown in Fig. L.16. We report the success rate of the open primitive on two oven objects using the pre-trained BC model conditional upon a successful grasp, as detailed in Table L.12.

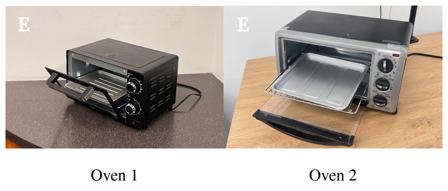


Figure L.16: Ovens used for evaluation

The "open" primitive struggles with handling oven objects because it only controls the velocity of the base in SE(2), and a stiff arm cannot exert sufficient force with a changing direction to operate the oven. To address this, we install a force sensor torque on the end-effector of the arm and apply impedance control, which provides passive compliance. This compliance allows the arm to align with the force direction of the articulation while the robot base moves backward, enhancing its effectiveness (Fig. L.15).

Open Primitive Success Rate		
	oven A	oven B
with FT-Sensor	5/5	5/5
w/o FT-Sensor	0/5	1/5

Table L.12: Effect of using Force Sensors.

L.5.4 Model Parameterization Details

We introduce a hierarchical policy framework that includes a primitive classifier, denoted as the high-level, and a conditional policy network, denoted as the low-level policy. The high-level policy takes as input cropped RGB images and produces logits that parameterize a discrete categorical distribution P_D. The low-level policy takes the same image as well as a discrete sample from distribution P_D, and produces the mean and standard deviation of a Gaussian distribution, which parameterizes the continuous parameter distribution P_C. In this section, we provide further details on the parameterization of the policy model. Additionally, we provide a list of all related hyperparameters in Table L.13.

Hyperparameters Table		
Hyperparameter	Symbol	Value
Mobile Manipulation Primitives Sequence	N	2
Number of Mobile Manipulation Primitives	N_p	4
Number of parameter dimension	M	3
BC Learning Rate	lr_{BC}	$1e-3$
Online Adaptation Learning Rate	lr_{Adp}	$1e-4$
Constrained mobile manipulation primitive execution Time	T	$2.5s$
Unlock velocity	v_{z}	$10cm/s$
Rotate velocity	v_{yaw}	$25 \circ /s$
Open velocity	V_{x}	$20cm/s$
Grasp offset	d	$2.5cm$
Batch size	B	16
Number of iteration during sample	N_{iter}	5
Number of rollout during sample	N_{rol}	5
Overall loss function hyperparameter	α	0.2

Table L.13: System hyperparameters

L.5.5 Hierarchical Open-loop Policy

In our setup, we introduce a hierarchical policy consisting of both a high-level policy and a low-level policy. The high-level policy takes visual inputs and outputs discrete actions, determining a sequence of primitive types. Meanwhile, the low-level policy takes both visual inputs and the discrete actions from the high-level policy, and outputs continuous parameters for the respective primitives. The high-level policy generates discrete actions, and the low-level policy outputs continuous parameters, together instantiating a sequence of primitives. The robot executes this sequence in an open-loop manner. To ensure that the action sequence produced by the open-loop policy maintains a fixed horizon, we have introduced a blank primitive during policy learning, which allows for the skipping of action execution.

L.5.6 Network Architecture and Policy Parameterization

Both the high-level and low-level policies share a frozen visual backbone, which is a ResNet-18 [157] pretrained on ImageNet [90]. We begin by cropping the handle from the RGB image and padding the cropped image into a square shape with zero-padding. The visual backbone takes this processed RGB image to output encoded visual features.
The high-level policy utilizes these encoded visual features to output a sequence

of N indices representing the types of constrained mobile manipulation primitives. For example, $[1, 3]$ corresponds to $[Unlock, Open]$. The high-level policy head is composed of a three-layer multi-layer perceptron (MLP) that outputs action logits sized $[B, N_p, N]$, where B is the batch size, N_p is the number of constrained mobile manipulation primitives, and N is the horizon of the mobile manipulation primitives sequence. A softmax layer then transforms these logits into probabilities for the categorical distribution. Discrete high-level actions are sampled from this distribution using simple greedy sampling.
The low-level policy head takes the encoded visual features and the sampled action from the high-level policy as input and outputs parameters sized $N + 1 \times M$, where $N + 1$ is the horizon of the primitive sequence (including both grasp and constrained mobile manipulation primitives), and M is the dimension of the low-level parameters. The grasp primitive uses an M-dimensional parameter, where M is 3, while the constrained mobile manipulation primitives use only a 1-dimensional parameter. To manage batch tensor computations across primitives with different parameter dimensions, the low-level policy outputs a "one size fits all" distribution over the parameters. For execution, similar to Maple [274], the grasp primitive utilizes the M-dimension parameter, but the other primitives only use the first dimension of this parameter. In our implementation, the low-level policy head generates the mean and standard deviation of Gaussian distributions for the primitives' low-level parameters. This head consists of a shared two-layer MLP, followed by separate fully connected layers for calculating the mean and the standard deviation. Specifically, the mean is processed through a third fully connected layer and then passed through a tanh activation function, while the standard deviation is processed through its own third fully connected layer and passed through a sigmoid activation function. Actions are then sampled from these Gaussian distributions and clipped to a range from -1 to 1.

L.5.7 Pretraining-Finetuning-Testing

We start with BC pre-training, we employ an Adam [195] optimizer with a learning rate of $1e - 3$ and denote this pre-trained BC model as the base model, π_b. It takes around 40 minutes to train the BC model using a 3090 Ti GPU. Following BC pre-training, we fine-tune the BC model π_b using both the BC training data and the online sample data collected on the testing object. We obtain the fine-tuned model π_f by using an Adam [195] optimizer, an overall loss function hyperparameter α of 0.2, and a learning rate of $1e - 4$. As detailed in Algorithm 7, we update the latest π_f after every N_{rol} rollouts. Collecting 25 rollouts takes around 45 minutes. And fine-tuning model takes around 10 minutes using a 3090 Ti GPU. We evaluate both the BC model and the fine-tuned model on the testing object.

Hardware features comparison						
	Arm payload	DoF arm	omni-base	footprint	base max speed	price
Stretch RE1 [439]	1.5kg	2	×	34 cm, 33 cm	0.6 m/s	20k USD
Go1-air + WidowX 250s [115]	0.25kg	6	✓	59 cm, 22 cm	2.5 m/s	10k USD
Franka + Clearpath Ridgeback [194]	3kg	7	✓	96 cm, 80 cm	1.1 m/s	75k USD
Franka + Omron LD-60 [316]	3kg	7	×	70 cm, 50 cm	1.8 m/s	50k USD
Xarm-6 + Agilex Ranger mini 2 (ours)	5kg	6	✓	74 cm, 50 cm	2.6 m/s	25k USD

Table L.14: Comparison of different aspects of popular hardware systems for mobile manipulation

L.5.8 Hardware details

Among the commercially available options, we found the Ranger Mini 2 from AgileX to be an ideal choice for robot base due to its stability, omni-directional velocity control, and high payload capacity to mount robot arms. The system uses an xArm-6 for manipulation, which is an effective low-cost arm with a high payload (5kg), and is widely accessible for research labs. The system uses a Nvidia Jetson computer to support real-time communication between sensors, the base, the arm, as well as a server that hosts large models. We use a D435 Intel Realsense camera mounted on the frame to collect RGBD images as ego-centric observations and a T265 Intel Realsense camera to provide odometry with visual slam, which is critical for resetting the robot when performing trials for Reinforcement Learning. The gripper is equipped with a 3D-printed hooker and an anti-slip tape. The overall cost of the entire system is around $25,000$ USD, making it an affordable solution for most robotics labs. We compare key aspects of our modular platform with that of other mobile manipulation platforms in Table L.14. This comparison highlights advantages of our system such as cost-effectiveness, reactivity, ability to support a high-payload arm, and a base with omnidirectional drive.

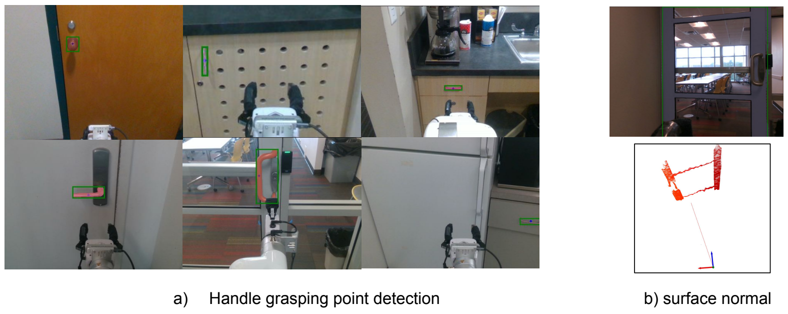


Figure L.17: **Detection robustness analysis:** a) When multiple handles are present in a scene, the detection model often fails to accurately identify the specific handle of interest. b) Visualizes the surface normal estimation process.

L.5.9 BC Model Failures

Generalizing to unseen objects with a pre-trained BC model poses significant challenges. As we discussed before, when encountering visually ambiguous objects, it is difficult to discern whether a door should be ’pulled’ or ’pushed’ using only visual observation. Additionally, if the training data exclusively features ’pull’ doors, the BC model may find it more challenging to articulate ’push’ doors, which it has never encountered before. Thanks to continual practice, RL can explore actions that deviate from those in the BC pre-training dataset and learn from these experiences. Consequently, this enables the robot to learn how to manipulate novel objects.

L.5.10 Detection Model Failures

The success of a door-opening task relies heavily on the initial accuracy of handle detection. However, we encounter several challenges in specific scenarios. As illustrated in the second row of the rightmost figure in Fig. L.17. when multiple handles are present in a scene, the detection model fails to accurately identify the specific handle of interest. We plan to address this issue in future work. Additionally, the performance of depth sensors can be compromised by lighting conditions, leading to grasping failures.
We also observed that for doors with knobs, as shown in Figure 8.7, it is challenging to simply use detection and segmentation models to provide an ideal grasping pose. Online adaptation enables the robot to adjust the grasp pose through real-world interactions, which significantly increases the success rate.

L.5.11 Robustness VLM Reward Model

In the main experiment, we conducted experiments to validate the effectiveness of the autonomous online adaptation with VLM reward models on only two objects. Additionally, we discovered that the CLIP model [308] demonstrates high robustness in classifying doors as open or closed across all the doors we tested and trained on.

L.5.12 Payload Limitation

Real-world door-opening tasks are challenging, as we discovered in our experiments where hardware proved to be one of the major bottlenecks. Most robotic arms, even including the xarm-6 with its 5 kg end-effector payload, are insufficient for opening extremely heavy doors. Moreover, the force torque sensors we utilized have a payload capacity of only 1 kg, significantly restricting their effectiveness in real-world door-opening applications.

L.5.13 Hardware Capability

To successfully operate various articulated objects, the first question we need to answer is: What is the right hardware platform for opening a real door? We compare against a different popular mobile manipulation system, namely the Stretch RE1 (Hello Robot). We test the ability of the robots to be teleoperated by a human expert to open two doors from different categories, specifically lever and knob doors. Each object was subjected to five trials. As shown in Table L.18, the outcomes of these trials revealed a significant limitation of the Stretch RE1: its payload capacity is inadequate for opening a real door, even when operated by an expert, while our system succeeds in all trials.

Expert teleoperation success rate		
	lever B	knob A
Stretch RE1	0/5	0/5
Ours	5/5	5/5

Figure L.18: **Hardware Comparison:** Human expert teleoperation success rate.

L.6 Bimanual Dexterity for Complex Tasks

L.6.1 Detailed Cost Analysis

Please see Table L.15 and Table L.16 for a detailed Bill of Materials and breakdown of the cost to create BiDex. While we assert that BiDex is low cost, we acknowledge that it is still not affordable for everyone such as hobbyists. We believe that the price of motion capture gloves will continue to decrease over time as technology improves and demand increases in our field as well as other adjacent fields.

Object	Quantity	Total
Manus Meta Glove	1	\$6000
Dynamixel XL330-M288	12	\$300
U2D2 Control PCB	1	\$20
5v 20A Power Supply	2	\$25
14 AWG Cabling	1	\$20
PLA Printer Plastic	N/A	\$10
Total		\$6375

Table L.15: We present the bill of materials of BiDex for two arms and hands. The total cost is around \$6000, mostly due to the Manus Meta gloves.

Object	Quantity	Total
xArm 6	2	\$18000
Ubuntu Laptop	1	\$2000
Mobile Base	1	\$6000
Zed Camera	3	\$1200
LEAP Hand or DLA Hand	2	\$4000
Total		\$31,2000

Table L.16: We present the bill of materials of the mobile robot setup. The robot and BiDex costs around \$35,000 which we believe is reasonable for a dexterous bimanual robot hand setup with 50+ degrees of freedom.

L.6.2 Assembly Instructions and Software

The assembly instructions are available on the project website. The hardware system and software will be a useful to recreate BiDex and create variants of it using high quality motion capture gloves. Our high quality teacher arm teleoperation is based off of [426] but the strength is increased to allow for the weight of the gloves and its mounting system.

L.6.3 About Manus Glove

We use the Manus Meta Quantum Metagloves [247] which is an $6000 tracking Mocap glove. Each finger is tracked by the glove and returns the fingertip positions as xyz-quaterion and also 4 different angles for each finger $\theta_{\mathrm{MCP}_{\mathrm{side}}}, \theta_{\mathrm{MCP}_{\mathrm{fwd}}}, \theta_{\mathrm{PIP}}, \theta_{\mathrm{DIP}}$ using hall effect sensors with very high accuracy and at 120hz. We use their Windows API (Linux is not available at time of release) and will release our version of that which sends the software to a Linux machine running ROS. These gloves are available for purchase at `https://www.manus-meta.com/`.

L.6.4 SteamVR Baseline

For the wrist tracking SteamVR baseline, we use the Manus Meta SteamVR trackers which connect to the gloves and seamlessly route the data through the aforementioned Windows API. They are wireless but require SteamVR Lighthouses setup around the perimeter of the workspace. In our test we mount the 4 SteamVR trackers on the ceiling to avoid as many occlusions as possible. We also mount the 4 trackers in a 16ft square around where the teleoperator would stand which is the recommended configuration. We will release this code for others to recreate in their comparison study.

L.6.5 Apple Vision Pro Baseline

The Apple Vision Pro baseline is based off of [282]. With this data, we control the hand using the same inverse kinematics as with the Manus Glove. For the arm, we scale, translate and rotate for the robot embodiment and then pass through inverse kinematics to control the arm.

L.6.6 Behavior Cloning Policy Architecture and Hyperparameters

We illustrate our policy architecture in Figure L.19. Our behavior cloning policy takes as input a RGB image and current hand joint angles (proprioception). We obtain tokens for the image observation via a ViT [94] and a token for joint proprioception via a linear layer. The weights of ViT is initialized from the Soup 1M model from [83]. The tokens then pass through a action chunking transformer [458], a encoder-decoder transformer, to output a sequence of actions.

Hyperparameter	Value
Behavior Policy Training	
optimizer	AdamW
base learning rate	3e-4
weight decay	0.05
optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.95$
batch size	64
learning rate schedule	cosine decay
total steps	10000
warmup steps	500
augmentation	GaussianBlur, Normalize, RandomResizedCrop
GPU	RTX4090 (24 gb)
Wall-clock time	$\sim$ 1 hour
Visual Backbone ViT Architecture	
patch size	16
#layers	12
#MHSA heads	12
hidden dim	768
class token	yes
positional encoding	sin cos
Action Chunking Transformer Architecture	
# encoder layers	6
# decoder layers	6
#MHSA heads	8
hidden dim	512
feedforward dim	2048
dropout	0.1
positional encoding	sin cos
action chunk	100

Table L.17: Hyperparameters for Behavior Cloning Policy Training

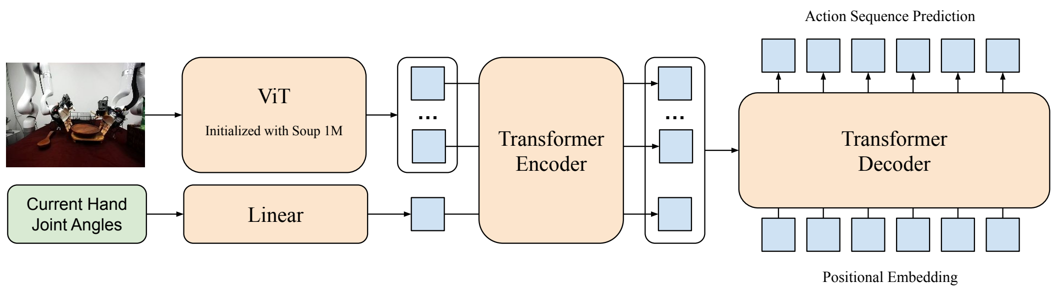


Figure L.19: Behavior Cloning Policy Architecture

The action space is the absolute joint angles of two arms and two hands. A key decision that greatly improves policy generalization is to exclude current arm joints from the proprioception. Intuitively, this may force the model to extract object information from image observations, rather than overfitting to predict actions close to current arm states.
We list key hyperparameters for our behavior policy training Table L.17. In general, we are able to obtain well-performing policies with 20-50 demonstrations and 1 hour of wall-clock time training on a RTX4090. With our easy-to-use teleoperation system, we are able to obtain diverse policies for complex bimanual dexterous tasks quickly.

L.7 NeuralMP: A Generalist Neural Motion Planner

L.7.1 Detailed Free Hand Motion Planning Results

In this section we perform additional analysis of the free hand motion planning results from the main Chapter. We include a more detailed version of Tab. 10.4 in the Appendix shown below (Tab. L.20). In this table, we additionally include the average (open loop) planning time per method and the average rate of safety violations. Safety violations are defined to occur where there are collisions, the robot hits its joint limits or there are torque limit errors. The open loop planning time for neural methods such as ours or MπNets involves simply measuring the total time taken for rolling out the policy and test time optimization (TTO). We find that sampling-based planners in general never collide when executed. If they produce a safety violation, it is only because they find a trajectory that is infeasible for the robot to execute on the hardware, due to joint or torque limit errors. Neural motion planning methods have much higher collision rates, though Neural MP has a significantly lower collision rate than MπNets, which we attribute to test-time optimization pruning out bad trajectories. We also note that not all collisions are created equal: some are slight, lightly grazing the environment objects while still achieving the goal, while others can be catastrophic, colliding heavily into the environment. In general, we found that our method tends to produce trajectories that may have slight collisions, though most of these are pruned out by TTO. With regards to planning time, MπNets is the fastest method, as our method expends additional compute rolling out 100x more trajectories and then selecting the best one using SDF-based collision checking.

	Bins (↑)	Shelf (↑)	Articulated (↑)	Avg. Success Rate (↑)	Avg. Planning Time (↓)	Avg. Safety Viol. Rate (↓)
Sampling-based Planning:						
`AIT*-80s` [380]	93.75	75	50.0	72.92	80	**0**
`AIT*-10s (fast)` [380]	75.0	37.5	25.0	45.83	10	2.1
Neural:						
Mπ`Nets` [108]	18.75	25.0	6.25	16.67	**1.0**	18.75
`Ours`	**100**	**100**	**87.5**	**95.83**	3.9	4.2

Figure L.20: Neural MP performs best across tasks for free-hand motion planning, demonstrating greater improvement as the task complexity grows.

L.7.2 Detailed In-hand Motion Planning Results

In this section, we extend the in-hand results shown in the main Chapter with additional baselines (AIT*-80s, AIT*-10s and MπNets). For this evaluation (see Tab. L.21, we consider two of the four in-hand motion planning objects, namely joystick and book. We find sampling-based methods are able to perform in-hand motion planning quite well, matching the performance of our base policy as well as our method without Objaverse data. We also see that MπNets is unable to perform in-hand motion planning on any of the evaluated tasks. This is likely because that network was not trained on data with objects in-hand, demonstrating the importance of including in-hand data when training neural motion planners. Finally, there is a significant gap in performance between our method with and without test-time optimization; pruning out colliding trajectories at test time is crucial for achieving high success rates on motion planning tasks.

	Book (↑)	Joystick (↑)	Avg. Success Rate (↑)	Avg. Planning Time (↓)	Avg. Safety Viol. Rate (↓)
Sampling-based Planning:					
AIT*-80s [380]	50	50	50	80	**0**
AIT*-10s (fast) [380]	25	50	37.5	10	**0**
Neural:					
MπNets [108]	0	0	0	1	37.5
Ours:					
Ours (no TTO)	25	**75**	50	**0.9**	50
Ours (no Objaverse)	50	50	50	3.9	50
Ours	**100**	**75**	**87.5**	3.9	12.5

Figure L.21: Neural MP performs best across tasks for in-hand motion planning, demonstrating greater improvement as the in-hand object becomes more challenging.

L.7.3 Test-time Optimization Analysis

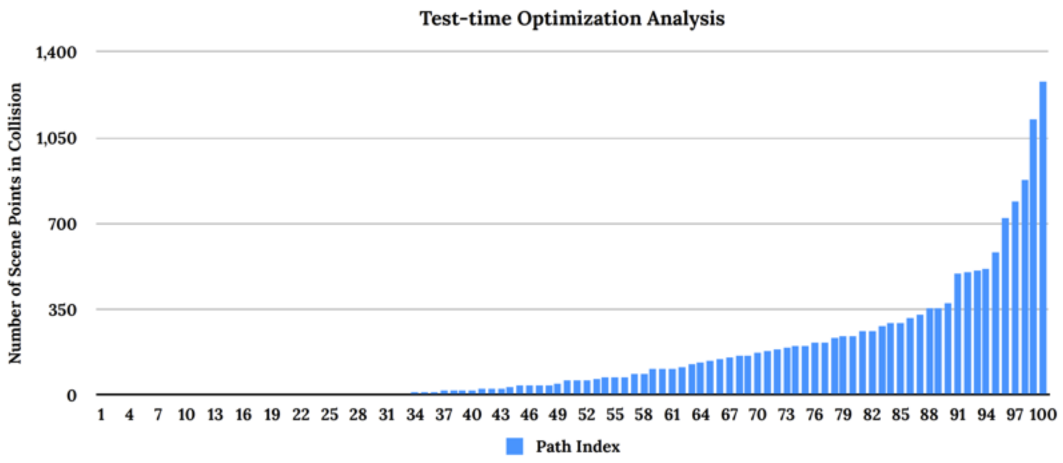


Figure L.22: **Test-time Optimization Analysis:** For the Bins Scene 1 task, we plot the number of points in collision across 100 sampled trajectories from the model. 25% of the trajectories are completely collision free and we select a trajectory execute from that subset.

To analyze what the test-time optimization procedure is doing, we first note that the base policy can sometimes produce slight collisions with the environment due to the imprecision of regression. As a result, when sampling from the policy, it is often likely that the policy will lightly graze objects which will count as failures when motion planning. We visualize a set of trajectories sampled from the policy here on our website for the real-world bins task. Observe that for some of the trajectories, the policy slightly intersects with the bin which would cause it to fail when executing in the real world, while for others it simply passes over the bin completely without colliding. We estimate the robot-scene intersection of all of these trajectories by comparing the robot SDF to the scene point-cloud and plot the range of values in Fig. L.22. We observe that 25% of trajectories do not collide with the environment, and we select for those. In principle, one could further optimize by selecting the trajectory that is furthest from the scene (using the SDF). In practice, we did not find this necessary and that selecting the first trajectory among those with the fewest expected collisions performed quite well in our experiments.

L.7.4 Ablations

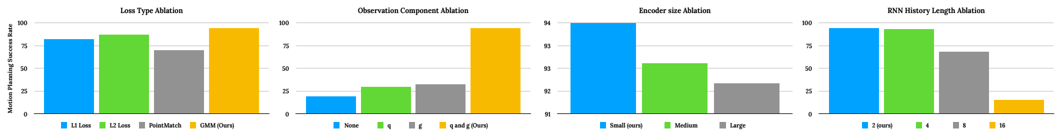


Figure L.23: **Ablation Results:** We evaluate four different components of Neural MP, loss type (*left*), observation components (*middle left*), encoder sizes (*middle right*), and RNN history length (*right*). We validate that our design decisions produce measurable improvements in motion planning success rates.

We run additional ablations analyzing components of our method in simulation using a subset of our dataset (100K trajectories) and include additional details for experiments discussed in the main Chapter.

Loss Types: For training objective, we evaluate 4 different options: GMM log likelihood (ours), MSE loss, L1 loss, and PointMatch loss (MπNets). PointMatch loss involves computing the l2 distance between the goal and the predicted end-effector pose using 1024 key-points. We plot the results on held out scenes in Fig. L.23. We find that GMM (ours) outperforms L2 loss, L1 loss, and PointMatch Loss (MπNets) by (7%, 12%, and 24%) respectively. One reason this may be the case is that sampling-based motion planners produce highly multi-modal trajectories: they can output entirely different trajectories for the same start and goal pair when sampled multiple times. Since Gaussian Mixture Models are generally more capable of capturing multi-modal distributions, they can hence fit our dataset well. At the same time, the PointMatch [108] loss struggles significantly on our data: it cannot distinguish between 0 and 180 degree flipped end-effector orientations, resulting in many failures due to incorrect end-effector orientations.

Observation Components: We evaluate whether our choice of observation components impacts the Neural MP's performance. In theory, the network should be able to learn as well from the point-cloud alone as when the proprioception is included, as the point-cloud contains a densely sampled point-cloud of the current and goal robot configurations. However, in practice, we find that this is not the case. Instead, removing either q or g or both severely harms performance as seen in Fig. L.23. We hypothesize that including the proprioception provides a richer signal for the correct delta action to take.

RNN History Length: In our experiments, we chose a history length of 2 for the RNN, after sweeping over values of 2, 4, 8, 16 based on performance. From Fig. L.23 we see history length 2 achieves the best performance at 94%, while using lengths 4, 8 and 16 achieve progressively decreasing success rates (92.67, 68, 14.67). One possible reason for this is that since point-clouds are already very dense representations that cover the scene quite well, the partial observability during training time is fairly low. A shorter history length also leads to faster training, due to smaller batches and fewer RNN unrolling steps.

Encoder Size: Finally, we briefly evaluate whether encoder size is important when training large-scale neural motion planners. We train 3 different size models: small (4M params), medium (8M params) and large (16M params). From the results in Fig. L.23, we find that the encoder size does not affect performance by a significant margin (94%, 93%, 92%) respectively and that the smallest model in fact performs best. Based on these results, we opt to use the small, 4M param model in our experiments.

Neural MP-MLP	**Neural MP-LSTM**	**Neural MP-Transformer**	**Neural MP-ACT**
65.0	82.5	**85.0**	47.5

Table L.18: Ablation of different architecture choices for the action decoder. We find that LSTMs and Transformers comparably while LSTMs boast faster inference times.

Architecture Ablation: In this experiment, we evaluate how different sequence modelling methods (Transformers and ACT [458]) and simpler action decoders such as MLPs compare against our design choice of using an LSTM. All methods are trained with the same dataset (of 1M trajectories), with the same encoder and GMM output distribution (with the exception of ACT which uses an L1 loss as per the ACT paper). We then evaluate them on held out motion planning tasks (Fig. L.18 which are replicas of our real-world tasks (Bins and Shelf). We note several findings: 1) ACT performs poorly, largely due to its design choice of using an L1 loss which prevents it from handling planner multi-modality effectively, 2) Neural MP with an MLP action decoder also performs significantly worse than LSTMs and Transformers, as it is unable to use history information effectively to reason about the next action 3) Transformers and LSTMs perform similarly, with the Transformer variant performing marginally better, but with significantly slower inference time (2x). Our method is amenable to any choice of sequence modeling architecture that performs well and has fast inference.

Neural MP-MotionBenchMaker	Neural MP-MπNets	Neural MP
0	32.5	82.5

Table L.19: Comparing different methods for generating datasets for motion planning. We find that policies trained on our data generalize best to held out scenes.

Dataset Ablation: Finally, we evaluate the quality of different dataset generation approaches for producing generalist neural motion planners. We do so by training policies on three different datasets (Neural MP, MπNets [108], and MotionBenchMaker [51]) and evaluated on held out motion planning tasks in simulation. We train each model to convergence for 10K epochs and then execute trajectories on two held out tasks that mirror our real world tasks: RealBins and RealShelf. For fairness, we do not include any Objaverse meshes in these tasks, since MPiNets and MotionBenchMaker only have primitive objects. Still, we find that our dataset performs best by a wide margin (Tab. L.19). In general, we found that policies trained on MotionBenchMaker do not generalize well. As mentioned in the related works section, this dataset lacks the realism and diversity necessary to train policies that can generalize to held out motion planning scenes.

L.7.5 Procedural Scene Generation

We formalize our procedural scene generation as a composition of randomly generated parameteric assets and sampled Objaverse meshes in Alg. 9
Objaverse sampling details: The Objaverse are sampled in the task-relevant sampling location of the programmatic asset(s) in the scene, such as between shelf rungs, inside cubbies or within cabinets. Similar to the programmatic assets, these Objaverse assets are also sampled from a category generator $X_{obj}(\mathbf{p})$. Here the parameter p specifies the size, position, orientation of the object as well as task-relevant sampling location of the object in the scene, such as between shelf rungs, inside cubbies or within cabinets. As discussed in the main Chapter, we propose an approach that iteratively adds assets to a scene by adjusting their position using the effective collision normal vector, computed from the existing assets in the scene. We detail the steps for doing this in Alg. 9.

L.7.6 Motion Planner Experts

We use three techniques to improve the data generation throughput when imitating motion planners at scale.
Hindsight Relabeling: Tight-space to tight-space problems are the most challenging, particularly for sampling-based planners, often requiring significant planning time (up to 120 seconds) for the planner to find a solution. For some problems, the expert planner is unable to find an exact solution and instead produces approximate solutions. Instead of discarding these, note that we use a goal-conditioned imitation

learning framework, where we can simply execute the trajectories in simulation and relabel the observed final state as the new goal.
Reversibility: We further improve our data generation throughput by observing that since motion planners inherently produce collision-free paths, the process is reversible, at least in simulation. This allows us to double our data throughput by reversing expert trajectories and re-calculating delta actions accordingly. Additionally, for a neural motion planner to be useful for practical manipulation tasks, it must be able to generate collision free plans for the robot even when it is holding objects. To enable such functionality, we augment our data generation process with trajectories where objects are spawned between the grippers of the robot end effector. There are transformed along with the end-effector during planning in simulation. We consider the object as part of the robot for collision checking and for the sake of our visual observations. In order to handle diverse objects that the robot might have to move with at inference time, we perform significant randomization of the in-hand object that we spawn in simulation. Specifically, we sample this object from the primitive categories of boxes, cylinders or spheres, or even from Objaverse meshes of everyday articles. We randomize the scale of the object between 3 and 30 cm along the longest dimension, and sample random starting locations within a 5cm cube around the end-effector mid-point between grippers.
Smoothing: Importantly, we found that naively imitating the output of the planner performs poorly in practice as the planner output is not well suited for learning. Specifically, plans produced by AIT* often result in way-points that are far apart, creating large action jumps and sparse data coverage, making it difficult to for networks to fit the data. To address this issue, we perform smoothing using cubic spline interpolation while enforcing velocity and acceleration limits. The implementation from MπNets performs well in practice, smoothing to a fixed 50 timesteps with a max spacing of 0.1 radians. In general, we found that smoothing is crucial for learning performance as it ensures action limits for each time-step transition.

L.7.7 Data Pipeline Parameters and Compute

In Table L.20, we provide a detailed list of all the parameters used in generating the data to train our model.

Hyper-parameter	Value
General Motion Planning Parameters	
collision checking distance	1cm
tight space configuration ratio	50%
dataset size	1M trajectories
minimum motion planning time	20s
maximum motion planning time	80s

Hyper-parameter	Value
General Obstacle Parameters	
in hand object ratio	0.5
in hand object size range	[[0.03, 0.03, 0.03], [0.3, 0.3, 0.3]]
in hand object xyz range	[[-0.05, -0.05, 0.], [0.05, 0.05, 0.05]]
min obstacle size	0.1
max obstacle size	0.3
table dim ranges	[[0.6, 1], [1.0, 1.5], [0.05, 0.15]]
table height range	[-0.3, 0.3]
num shelves range	[0, 3]
num open boxes range	[0, 3]
num cubbys range	[0, 1]
num microwaves range	[0, 3]
num dishwashers range	[0, 3]
num cabinets range	[0, 3]
Objaverse Mesh Parameters	
scale range	[0.2, 0.4]
x pos range	[0.2, 0.4]
y pos range	[-0.4, 0.4]
number of mesh objects per programmatic asset	[0, 3]
number of mesh objects on the table	[0, 5]
Table Parameters	
width range	[0.8, 1.2]
depth range	[0.4, 0.6]
height range	[0.35, 0.5]
thickness range	[0.03, 0.07]
leg thickness range	[0.03, 0.07]
leg margin range	[0.05, 0.15]
position range	[[0, 0.8], [-0.6, 0.6]]
z axis rotation range	[0, 3.14]
Shelf Parameters	
width range	[0.5, 1]
depth range	[0.2, 0.5]
height range	[0.5, 1.2]
num boards range	[3, 5]

Hyper-parameter	Value
board thickness range	[0.02, 0.05]
backboard thickness range	[0.0, 0.05]
num vertical boards range	[0, 3]
num side columns range	[0, 4]
column thickness range	[0.02, 0.05]
position range	[[0, 0.8], [-0.6, 0.6]]
z axis rotation range	[-1.57, 0]
Open Box Parameters	
width range	[0.2, 0.7]
depth range	[0.2, 0.7]
height range	[0.3, 0.5]
thickness range	[0.02, 0.06]
front scale range	[0.6, 1]
position range	[[0.0, 0.8], [-0.6, 0.6]]
z axis rotation range	[-1.57, 0.0]
Cubby Parameters	
cubby left range	[0.4, 0.1]
cubby right range	[-0.4, 0.1]
cubby top range	[0.85, 0.35]
cubby bottom range	[0.0, 0.1]
cubby front range	[0.8, 0.1]
cubby width range	[0.35, 0.2]
cubby horizontal middle board z axis shift range	[0.45, 0.1]
cubby vertical middle board y axis shift range	[0.0, 0.1]
board thickness range	[0.02, 0.01]
external rotation range	[0, 1.57]
internal rotation range	[0, 0.5]
num shelves range	[3, 5]
Microwave Parameters	
width range	[0.3, 0.6]
depth range	[0.3, 0.6]
height range	[0.3, 0.6]
thickness range	[0.01, 0.02]
display panel width range	[0.05, 0.15]

Hyper-parameter	**Value**
distance range	[0.5, 0.8]
external z axis rotation range	[-2.36, -0.79]
internal z axis rotation range	[-0.15, 0.15]
Dishwasher Parameters	
width range	[0.4, 0.6]
depth range	[0.3, 0.4]
height range	[0.5, 0.7]
control panel height range	[0.1, 0.2]
foot panel height range	[0.1, 0.2]
wall thickness range	[0.01, 0.02]
opening angle range	[0.5, 1.57]
distance range	[0.6, 1.0]
external z axis rotation range	[-2.36, -0.79]
internal z axis rotation range	[-0.15, 0.15]
Cabinet Parameters	
width range	[0.5, 0.8]
depth range	[0.25, 0.4]
height range	[0.6, 1.0]
wall thickness range	[0.01, 0.02]
left opening angle range	[0.7, 1.57]
right opening angle range	[0.7, 1.57]
distance range	[0.6, 1.0]
external z axis rotation range	[-2.36, -0.79]
internal z axis rotation range	[-0.15, 0.15]

Table L.20: **Data Generation Hyper-parameters.** We provide a detailed list of hyper-parameters used to procedurally generate a vast variety of scenes in simulation.

Compute: To collect a vast data of trajectories, we parallelize data collection across a cluster of 2K CPUs. It takes around 3.5 days to collect 1M trajectories.

L.7.8 Network Training Details

We first describe additional details regarding our neural policy, and then discuss how it is trained. Following the design decisions of MπNets [108], we construct a segmented point-cloud for the robot, consisting of the robot point-cloud, the target goal robot point-cloud and the obstacle point-cloud. Here we note two key differences from MπNets: 1) our network conditioned on the target joint angles,

Hyper-parameter	**Value**

PointNet++ Architecture	`PointnetSAModule(`
	`    npoint=128,`
	`    radius=0.05,`
	`    nsample=64,`
	`    mlp=[1, 64, 64, 64],`
	`)`
	`PointnetSAModule(`
	`    npoint=64,`
	`    radius=0.3,`
	`    nsample=64,`
	`    mlp=[64, 128, 128, 256],`
	`)`
	`PointnetSAModule(`
	`    nsample=64,`
	`    mlp=[256, 512, 512],`
	`)`
	`MLP(`
	`    Linear(512, 2048),`
	`    GroupNorm(16, 2048),`
	`    LeakyReLU,`
	`    Linear(2048, 1024),`
	`    GroupNorm(16, 1024),`
	`    LeakyReLU,`
	`    Linear(1024, 1024)`
	`)`

LSTM	1024 hidden dim, 2 layers
Inputs	q_t, g, PCD_t
Batch Size	16
Learning Rate	0.0001
GMM	5 modes
Sequence Length (seq length)	2
Point Cloud Parameters	
Number of Robot / Goal Point-cloud Points	2048
Number of Obstacle Point-cloud Points	4096

Table L.21: Hyper-parameters for the model

while MπNets only does so through the segmented point-cloud, 2) we condition on the target joint angles, not end-effector pose, decisions that we found improved adherence to the overall target configuration. For in-hand motion planning, we extend this representation by considering the object in-hand as part of the robot for the purpose of segmentation. We include a hyper-parameter list for our neural motion planner in Table L.21. We train a 20M parameter neural network across our dataset of 1M trajectories. The PointNet++ encoder is 4M parameters and outputs an embedding of dimension 1024. We concatenate this embedding with the encoded q_t and g vectors and pass this into the 16M parameter LSTM decoder. The decoder outputs weights, means, and standard deviations of the 5 GMM modes. The model is trained with negative log likelihood loss for 4.5M gradient steps, which takes 2 days on a 4090 GPU with batch size of 16.

L.7.9 Real Robot Setup

Hardware: For all of our experiments, we use a Franka Emika Panda Robot, which is a 7 degree of freedom manipulator arm. We control the robot using the manimo library (https://github.com/AGI-Labs/manimo) and perform all of experiments using their joint position controller with the default PD gains. The robot is mounted to a fixed base pedestal behind a desk of size .762m by 1.22m with variable height. For sensing, we use four extrinsically calibrated depth cameras, Intel Realsense 435 / 435i, placed around the scene in order to accurately capture the environment. We project the depth maps from each camera into 3D and combine the individual point-clouds into a single scene representation. We then post-process the point-cloud by cropping it to the workspace, filtering outliers and denoising, and sub-sampling a set of 4096 points. This processed point-cloud is then used as input to the policy.

Algorithm 14 Open-Loop NeuralMP Execution

```
1: Input: Neural MP $\pi_\theta$, segmentor $\mathcal{S}$, initial angles $q_0$, scene point-cloud $PCD_{full}$, goal $g$, horizon $H$
2: Output: Executed trajectory on the robot
3: Initialize: Timestep $t \leftarrow 0$
4: Initialize: Trajectory $\tau \leftarrow \{\}$
5: $PCD_0 \leftarrow \mathcal{S}(PCD_{full}) \cup PCD_{q_0} \cup PCD_g$
6: while goal $g$ not reached and $t < H$ do
7:     $a_t \sim \pi_\theta(q_{t-1}, PCD_{t-1}, g)$
8:     $q_t \leftarrow q_{t-1} + a_t$
9:     $PCD_t \leftarrow (PCD_{t_1} \setminus PCD_{q_{t-1}}) \cup PCD_{q_t}$
10:    $t \leftarrow t + 1$
11:    $\tau \leftarrow \tau + a_t$
12: end while
13: Execute the $\tau$ open loop on the robot.
```

Representation Collision Checking and Segmentation: In order to perform real world collision checking and robot point-cloud segmentation, we require a representation of the robot to check intersections with the scene (collision checking) and to filter out robot points from the scene point-cloud (segmentation). While the robot mesh is the ideal candidate for these operations, it is far too slow to run in real time. Instead, we approximate the robot mesh as spheres (visualized in Fig. L.24) as we found this performs well in practice while operating an order of magnitude faster. We use 56 spheres in total to approximate the links of the robot as well as the end-effector and gripper. These have radii ranging from 2cm to 10cm and are defined relative to the center of mass of the link. This representation is a conservative one: it encapsulates the robot mesh, which is desirable for segmentation as this helps account for sensing errors which would place robot points outside of the robot mesh.

Robot Segmentation: In order to perform robot segmentation in the real world, we use the spherical representation to filter out robot points in the scene, so only the obstacle point-cloud remains. Doing so requires computing the Signed Distance Function (SDF) of the robot representation and then checking the scene point-cloud against it, removing points from the point-cloud in which SDF value is less than threshold ϵ. For the spherical representation, the SDF computation is efficient: for a sphere with center C and radius r, the SDF of point x is simply $||x - C||_2 - r$. In our experiments, we use a threshold ϵ of 1cm. We then replace the removed points with points sampled from the robot mesh of the robot. This is done by pre-sampling a robot point-cloud from the robot mesh at the default configuration, then performing forward kinematics using the current joint angles q_t and transforming the robot point-cloud accordingly. Replacing the real robot point-cloud with the sampled one ensures that the only difference between sim and real is the obstacle point-cloud.

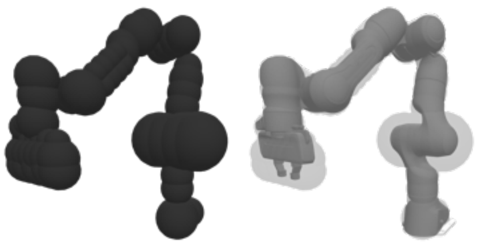


Figure L.24: We visualize the spherical representation on the left and overlay it on the robot mesh on the right.

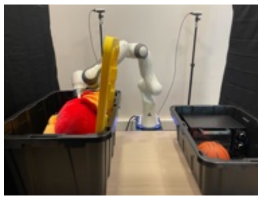
Real-world Collision Checking: Given the SDF, collision checking is also straightforward, we denote the robot in collision if any point in the scene point-cloud (this is after robot segmentation) has SDF value less than 1cm. Note this means that first state is by definition collision free. Also, this technique will not hold if performing closed loop planning, in that case this method would always denote the state as collision free as the points with SDF value less than 1cm would be segmented out for each intermediate point-cloud.

Open Loop Deployment: For open-loop execution of neural motion planners, we execute the following steps: 1) generate the segmented point-cloud at the first frame, 2) predict the next trajectory way-point by computing a forward pass through the network and sampling an action, 3) update the current robot point-cloud with mesh-sampled point-cloud at the predicted way-point, and 4) repeat until goal reaching success or maximum rollout length is reached. The entire trajectory is then executed on the robot after the rollout. Please see Alg. 14 for a more detailed description of our open-loop deployment method.

L.7.10 Tasks

Bins: This task requires the neural planner to perform collision avoidance when moving in-between, around and inside two different industrial bins pictured in the first row of Fig. L.25. We randomize the position and orientation of the bins over the table and include the following objects as additional obstacles for the robot to avoid: toaster, doll, basketball, bin cap, and white box. The small bin is of size 70cm x 50cm x 25cm. The larger bin is of size 70cm x 50cm x 37cm. The bins are placed at two sides of the table. Between tasks, we randomize the orientation of the bins between 0 and 45 degrees and we swap the bin ordering (which bin is on the left vs. the right). The bins are placed 45cm in front of the robot, and shifted 60cm left/right.
Shelf: This task tests the agent's ability to handle horizontal obstacles (the rungs of the shelf) while maneuvering in tighter spaces (row two in Fig. L.25). We randomize the size of the shelf (by changing the number of layers in the shelf from 3 to 2) as well as the position and orientation (anywhere at least .8m away from the robot) with 0 or 30 degrees orientation. The obstacles for this task include the toaster, basketball, baskets, an amazon box and an action figure which increase the difficulty. The shelf obstacle itself is of size 35cm x 80cm x 95cm.
Articulated: We extend our evaluation to a more complex primary obstacle, the cabinet, which contains one drawer and two doors and tight internal spaces with small cubby holes (row three of Fig. L.25). We randomize the position of the entire cabinet over the table, the joint positions of the drawer and doors and the sizes of the cubby holes. The obstacles for this task are xbox controller box, gpu, action figure, food toy, books and board game box. The size of the cabinet is 40cm x 75cm x 80cm. The size of the top drawer is 30cm x 65cm x 12cm. The size of the cubbies is 35cm x 35cm x 25cm. The drawer has an opening range of 0-30cm and the doors open between 0 and 180 degrees.
In-Hand Motion Planning: In this task (shown in row four of Fig. L.25), the planner needs to reason about collisions with not only the robot and the environment, but the held object too. We initialize the robot with an object grasped in-hand and run motion planning to reach a target configuration. For this task, we fix the obstacle (shelf) and its position (directly 80cm in front of the robot), instead randomizing across in-hand objects and configurations. We select four objects that vary significantly in size and shape: Xbox controller (18cm x 15cm x 8cm), book (17cm x 23cm x 5cm), toy sword (65cm x 10cm x 2cm), and board game (25cm x 25cm x 6cm). For this evaluation, we assume the object is already grasped by the robot, and the robot must just move with the object in-hand while maintaining its grasp.

(a) Bins Scene 1


(b) Bins Scene 2

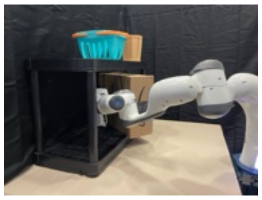
(c) Bins Scene 3

(d) Bins Scene 4

(e) Shelf Scene 1

(f) Shelf Scene 2

(g) Shelf Scene 3


(h) Shelf Scene 4

(i) Articulated Scene 1

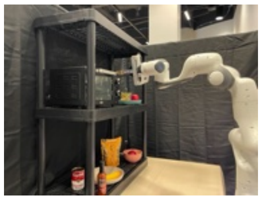
(j) Articulated Scene 2

(k) Articulated Scene 3

(l) Articulated Scene 4

(m) In Hand Object 1

(n) In Hand Object 2


(o) In Hand Object 3

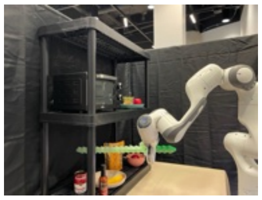

(p) In Hand Object 4

Figure L.25: Images of our 16 evaluation scenes.

L.7.11 Perception Visualization and Analysis

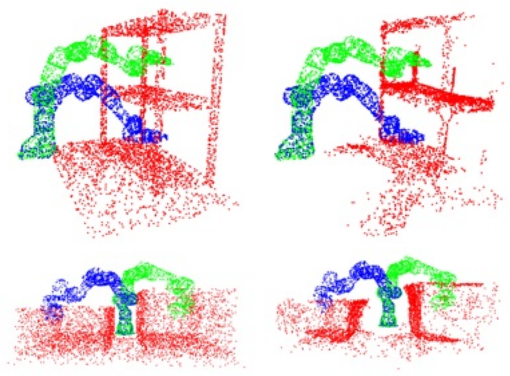


Figure L.26: **Visualization of Sim and Real point-clouds**: We visualize point-clouds of the Bins and Shelf task in sim and real, in the same poses. Due to noise in depth sensing, the real world point-clouds have significantly more deformations, yet our policy generalizes well to these tasks.

We compare point-clouds from simulation and the real world for the Bins and Shelf task and analyze their properties. We replicate Bins Scene 4 and Shelf Scene 1 in simulation: simply measure the dimensions and positions of the real world objects and set those dimensions in simulation using the OpenBox and Shelf procedural assets. As seen in Fig. L.26, simulated point-clouds are far cleaner than those in the real world, which are noisy and perhaps more importantly, partial. The real-world point-clouds often have portions missing due to camera coverage as for large objects it is challenging to cover the scene well while remaining within the depth camera operating range. However, we find that our policy is still able to able operate well in these scenes, as PointNet++ is capable of handling partial point-clouds and is trained on a diverse dataset containing many variations of boxes and shelves with different types and number of components as well as sizes, which may enable the policy to generalize to partial boxes and shelves observed in the real world.

Bibliography

[1] Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018. 23

[2] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. In *Conference on robot learning*, pages 403–415. PMLR, 2023. 123, 137

[3] Ananye Agarwal, Shagun Uppal, Kenneth Shaw, and Deepak Pathak. Dexterous functional grasping. In *Conference on Robot Learning*, pages 3453–3467. PMLR, 2023. 117

[4] AgileX Robotics. Ranger mini. `https://global.agilex.ai/products/ranger-mini`, 2023. Omnidirectional mobile robot platform. 116

[5] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *Conference on Robot Learning*, 2022. 87, 99

[6] OpenAI: 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, 85, 111, 137

[7] Brandon Amos, Ivan Dario Jimenez Rodriguez, Jacob Sacks, Byron Boots, and J. Zico Kolter. Differentiable mpc for end-to-end planning and control. In *NeurIPS*, 2018. 41

[8] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *NeurIPS*, 2017. 9, 14, 21, 23, 27, 46, 140, 143

[9] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *IJRR*, 2020. 54

[10] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009. 45

[11] Philip Arm, Mayank Mittal, Hendrik Kolvenbach, and Marco Hutter. Pedipulate: Enabling manipulation skills using a quadruped robot's leg. In *41st IEEE Conference on Robotics and Automation (ICRA 2024)*, 2024. 87

[12] Sridhar Pandian Arunachalam, Sneha Silwal, Ben Evans, and Lerrel Pinto. Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation. *arXiv preprint arXiv:2203.13251*, 2022. 41, 55

[13] Sridhar Pandian Arunachalam, Irmak Güzey, Soumith Chintala, and Lerrel Pinto. Holo-dex: Teaching dexterity with immersive mixed reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5962–5969. IEEE, 2023. 112

[14] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15619–15629, 2023. 137

[15] Christopher G Atkeson, PW Babu Benzun, Nandan Banerjee, Dmitry Berenson, Christoper P Bove, Xiongyi Cui, Mathew DeDonato, Ruixiang Du, Siyuan Feng, Perry Franklin, et al. What happened at the darpa robotics challenge finals. *The DARPA robotics challenge finals: Humanoid robots to the rescue*, pages 667–684, 2018. 98, 100

[16] Mohammad Babaeizadeh, Mohammad Taghi Saffar, Suraj Nair, Sergey Levine, Chelsea Finn, and Dumitru Erhan. Fitvid: Overfitting in pixel-level video prediction. *arXiv preprint arXiv:2106.13195*, 2021. 154

[17] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017. 54

[18] Shikhar Bahl, Mustafa Mukadam, Abhinav Gupta, and Deepak Pathak. Neural dynamic policies for end-to-end sensorimotor learning. In *NeurIPS*, 2020. 41, 54

[19] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Human-to-robot imitation in the wild. *RSS*, 2022. 27, 29, 41, 45, 46, 49, 57, 97, 103

[20] Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from human videos as a versatile representation for robotics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13778–13790, 2023. 4, 7, 53, 57, 99, 135

[21] 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. 87

[22] Philip J Ball, Jack Parker-Holder, Aldo Pacchiano, Krzysztof Choromanski, and Stephen Roberts. Ready policy one: World building through active learning. In *International Conference on Machine Learning*, pages 591–601. PMLR, 2020. 14

[23] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient Online Reinforcement Learning with Offline Data. In *ICML*, 2023. 91, 156

[24] Nandan Banerjee, Xianchao Long, Ruixiang Du, Felipe Polido, Siyuan Feng, Christopher G Atkeson, Michael Gennert, and Taskin Padir. Human-supervised control of the atlas humanoid robot for traversing doors. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 722–729. IEEE, 2015. 98, 100

[25] Aayush Bansal, Bryan Russell, and Abhinav Gupta. Marr revisited: 2d-3d alignment via surface normal prediction. In *CVPR*, 2016. 40

[26] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. V-jepa: Latent video prediction for visual representation learning. 2023. 68, 73, 76, 77

[27] 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. 87

[28] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *NeurIPS*, pages 1471–1479, 2016. 3, 13, 27, 46, 140

[29] Gedas Bertasius, Hyun Soo Park, Stella X Yu, and Jianbo Shi. First person action-object detection with egonet. *arXiv preprint arXiv:1603.04908*, 2016. 40

[30] Lucas Beyer, Damien Vincent, Olivier Teboul, Sylvain Gelly, Matthieu Geist, and Olivier Pietquin. Mulex: Disentangling exploitation from exploration in deep rl. *arXiv preprint arXiv:1907.00868*, 2019. 13

[31] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023. 5, 68, 69, 70, 76, 77

[32] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 69, 70, 73, 75, 76

[33] R. Bohlin and L.E. Kavraki. Path planning using lazy prm. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 521–528 vol.1, 2000. doi: 10.1109/ROBOT.2000.844107. 7, 122

[34] Jonathan Bohren, Radu Bogdan Rusu, E Gil Jones, Eitan Marder-Eppstein, Caroline Pantofaru, Melonee Wise, Lorenz Mösenlechner, Wim Meeussen, and Stefan Holzer. Towards autonomous robotic butlers: Lessons learned with the pr2. In *2011 IEEE International Conference on Robotics and Automation*, pages 5568–5575. IEEE, 2011. 116

[35] Gerald Brantner and Oussama Khatib. Controlling ocean one: Human–robot collaboration for deep-sea manipulation. *Journal of Field Robotics*, 38(1):28–51, 2021. 112

[36] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. 1, 123

[37] 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. *RSS*, 2023. 1, 54, 87, 97, 99, 123

[38] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023. 5, 68, 69

[39] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. URL `https://openai.com/research/video-generation-models-as-world-simulators`. 1, 2, 5, 136, 137

[40] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 2020. 1, 41, 53, 135, 136

[41] Arthur E Bryson and Yu-Chi Ho. *Applied optimal control: optimization, estimation, and control*. Routledge, 2018. 45

[42] Bernadette Bucher, Karl Schmeckpeper, Nikolai Matni, and Kostas Daniilidis. Adversarial curiosity. *arXiv preprint arXiv:2003.06082*, 2020. 13

[43] Lars Buesing, Theophane Weber, Sebastien Racaniere, SM Eslami, Danilo Rezende, David P Reichert, Fabio Viola, Frederic Besse, Karol Gregor, Demis Hassabis, et al. Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*, 2018. 12

[44] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018. 13, 140

[45] Samuel R Buss and Jin-Su Kim. Selectively damped least squares for inverse kinematics. *Journal of Graphics tools*, 10(3):37–49, 2005. 114

[46] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics*, 2007. 41

[47] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268, 2017. 119

[48] Víctor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giró-i Nieto, and Jordi Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, pages 1317–1327. PMLR, 2020. 23

[49] Zhe Cao, Ilija Radosavovic, Angjoo Kanazawa, and Jitendra Malik. Reconstructing hand-object interactions in the wild. In *CVPR*, pages 12417–12426, 2021. 41

[50] Joao Carvalho, An T Le, Mark Baierl, Dorothea Koert, and Jan Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1916–1923. IEEE, 2023. 124

[51] Constantinos Chamzas, Carlos Quintero-Pena, Zachary Kingston, Andreas Orthey, Daniel Rakita, Michael Gleicher, Marc Toussaint, and Lydia E Kavraki. Motionbenchmaker: A tool to generate and benchmark motion planning datasets. *IEEE Robotics and Automation Letters*, 7(2):882–889, 2021. 124, 178

[52] Angel Chang, Manolis Savva, and Christopher D Manning. Learning spatial knowledge for text to 3d scene generation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 2028–2038, 2014. 123

[53] Angel Chang, Will Monroe, Manolis Savva, Christopher Potts, and Christopher D Manning. Text to 3d scene generation with rich lexical grounding. *arXiv preprint arXiv:1505.06289*, 2015. 123

[54] Matthew Chang, Theophile Gervet, Mukul Khanna, Sriram Yenamandra, Dhruv Shah, So Yeon Min, Kavit Shah, Chris Paxton, Saurabh Gupta, Dhruv Batra, et al. Goat: Go to any thing. *arXiv preprint arXiv:2311.06430*, 2023. 97

[55] Yevgen Chebotar, Karol Hausman, Yao Lu, Ted Xiao, Dmitry Kalashnikov, Jake Varley, Alex Irpan, Benjamin Eysenbach, Ryan Julian, Chelsea Finn, et al. Actionable models: Unsupervised offline reinforcement learning of robotic skills. *arXiv preprint arXiv:2104.07749*, 2021. 23, 140

[56] Annie S Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from" in-the-wild" human videos. *arXiv preprint arXiv:2103.16817*, 2021. 27, 41, 57

[57] Haoxin Chen, Yong Zhang, Xiaodong Cun, Menghan Xia, Xintao Wang, Chao Weng, and Ying Shan. Videocrafter2: Overcoming data limitations for high-quality video diffusion models, 2024. 75, 76

[58] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84):eadc9244, 2023. 127

[59] Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized Ensembled Double Q-Learning: Learning Fast Without a Model. In *ICLR*, 2021. 91

[60] Xuxin Cheng, Ashish Kumar, and Deepak Pathak. Legs as manipulator: Pushing quadrupedal agility beyond locomotion. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5106–5112. IEEE, 2023. 87

[61] Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme parkour with legged robots. *ICRA*, 2024. 85, 122, 123

[62] Cheng Chi, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Iterative residual policy: for goal-conditioned dynamic manipulation of deformable objects. *The International Journal of Robotics Research*, page 02783649231201201, 2022. 99

[63] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *RSS*, 2023. 1, 112

[64] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *RSS*, 2024. 113

[65] Sachin Chitta, Benjamin Cohen, and Maxim Likhachev. Planning for autonomous door opening with a mobile manipulator. In *2010 IEEE International Conference on Robotics and Automation*, pages 1799–1806. IEEE, 2010. 98, 100

[66] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 12

[67] Jongwook Choi, Archit Sharma, Sergey Levine, Honglak Lee, and Shixiang (Shane) Gu. Variational empowerment as representation learning for goal-based reinforcement learning. In *Deep Reinforcement Learning workshop at the Conference on Neural Information Processing Systems (DRL)*, 2020. 18, 23

[68] Sammy Christen, Wei Yang, Claudia Pérez-D'Arpino, Otmar Hilliges, Dieter Fox, and Yu-Wei Chao. Learning human-to-robot handovers from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 127

[69] Vivian Chu, Tesca Fitzgerald, and Andrea L Thomaz. Learning object affordances by leveraging the combination of human-guidance and self-exploration. In *International Conference on Human-Robot Interaction*, 2016. 41

[70] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, pages 4754–4765, 2018. 55

[71] Kevin Clark, Paul Vicol, Kevin Swersky, and David J Fleet. Directly fine-tuning diffusion models on differentiable rewards. *arXiv preprint arXiv:2309.17400*, 2023. 68, 69, 70, 72

[72] Mark Cutler, Thomas J Walsh, and Jonathan P How. Reinforcement Learning with Multi-Fidelity Simulators. In *ICRA*, pages 3888–3895. IEEE, 2014. 85

[73] Murtaza Dalal, Deepak Pathak, and Russ R Salakhutdinov. Accelerating robotic reinforcement learning via parameterized action primitives. *NeurIPS*, 2021. 41, 54

[74] Murtaza Dalal, Ajay Mandlekar, Caelan Garrett, Ankur Handa, Ruslan Salakhutdinov, and Dieter Fox. Imitating task and motion planning with visuomotor transformers. In *Conference on Robot Learning*, 2023. 123, 127

[75] Murtaza Dalal, Jiahui Yang, Russell Mendonca, Youssef Khaky, Ruslan Salakhutdinov, and Deepak Pathak. Neural mp: A generalist neural motion planner. *arXiv preprint arXiv:2409.05864*, 2024. 7

[76] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018. VIII, 40, 41, 58, 59, 62, 145, 147, 151, 152, 153, 155

[77] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, , Antonino Furnari, Jian Ma, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *IJCV*, 2022. 40, 41

[78] Christian Daniel, Gerhard Neumann, Oliver Kroemer, and Jan Peters. Hierarchical relative entropy policy search. *Journal of Machine Learning Research*, 2016. 54

[79] Michael Danielczuk, Arsalan Mousavian, Clemens Eppner, and Dieter Fox. Object rearrangement using learned implicit collision functions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6010–6017. IEEE, 2021. 133

[80] Ahmad Darkhalil, Dandan Shan, Bin Zhu, Jian Ma, Amlan Kar, Richard Higgins, Sanja Fidler, David Fouhey, and Dima Damen. Epic-kitchens visor benchmark: Video segmentations and object relations. In *NeurIPS Track on Datasets and Benchmarks*, 2022. 40

[81] Abhishek Das, Satwik Kottur, José M.F. Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *ICCV*, 2017. 41

[82] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019. 53, 112

[83] Sudeep Dasari, Mohan Kumar Srirama, Unnat Jain, and Abhinav Gupta. An unbiased look at datasets for visuo-motor pre-training. In *Conference on Robot Learning*, pages 1183–1198. PMLR, 2023. 119, 171

[84] Todor Davchev, Kevin Sebastian Luck, Michael Burke, Franziska Meier, Stefan Schaal, and Subramanian Ramamoorthy. Residual learning from demonstration. *arXiv preprint arXiv:2008.07682*, 2020. 41

[85] Mathew DeDonato, Felipe Polido, Kevin Knoedler, Benzun PW Babu, Nandan Banerjee, Christoper P Bove, Xiongyi Cui, Ruixiang Du, Perry Franklin, Joshua P Graff, et al.

Team wpi-cmu: achieving reliable humanoid behavior in the darpa robotics challenge. *Journal of Field Robotics*, 34(2):381–399, 2017. 98, 100

[86] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *International Conference on machine learning (ICML)*, 2011. 55

[87] Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Found. Trends Robot*, 2013. 55

[88] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. Procthor: Large-scale embodied ai using procedural generation. *Advances in Neural Information Processing Systems*, 35:5982–5994, 2022. 123

[89] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 123, 124, 126

[90] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848. 105, 165

[91] Eadom Dessalene, Chinmaya Devaraj, Michael Maynord, Cornelia Fermuller, and Yiannis Aloimonos. Forecasting action through contact representations from first person video. *TPAMI*, 2021. 40

[92] Runyu Ding, Yuzhe Qin, Jiyue Zhu, Chengzhe Jia, Shiqi Yang, Ruihan Yang, Xiaojuan Qi, and Xiaolong Wang. Bunny-visionpro: Bimanual dexterous teleoperation with real-time retargeting using vision pro. 2024. 111, 112, 116

[93] Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment, 2023. 69

[94] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 171

[95] Anca D Dragan, Nathan D Ratliff, and Siddhartha S Srinivasa. Manipulation planning with goal sets using constrained trajectory optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 4582–4588. IEEE, 2011. 122

[96] Michael O'Gordon Duff and Andrew Barto. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. PhD thesis, University of Massachusetts at Amherst, 2002. 27

[97] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018. 10, 23, 53, 55, 140

[98] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *RSS*, 2022. 41, 53, 153

[99] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019. 3, 10, 23, 140

[100] D Eigen and R Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. corr, abs/1411.4734. *arXiv preprint arXiv:1411.4734*, 2014. 40

[101] Sarah Elliott, Zhe Xu, and Maya Cakmak. Learning generalizable surface cleaning actions from demonstration. In *International Symposium on Robot and Human Interactive Communication*, 2017. 41

[102] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018. 18, 23, 27, 33, 140

[103] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *NeurIPS*, 2019. 23

[104] Yuxin Fang, Bencheng Liao, Xinggang Wang, Jiemin Fang, Jiyang Qi, Rui Wu, Jianwei Niu, and Wenyu Liu. You only look at one sequence: Rethinking transformer in vision through object detection. *CoRR*, abs/2106.00666, 2021. URL `https://arxiv.org/abs/2106.00666`. 73, 76, 77

[105] Yazan Farha, Alexander Richard, and Juergen Gall. When will you do what?-anticipating temporal occurrences of activities. In *CVPR*, 2018. 41

[106] Weixi Feng, Xuehai He, Tsu-Jui Fu, Varun Jampani, Arjun Akula, Pradyumna Narayana, Sugato Basu, Xin Eric Wang, and William Yang Wang. Training-free structured diffusion guidance for compositional text-to-image synthesis, 2023. 69

[107] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2786–2793. IEEE, 2017. 23, 55

[108] Adam Fishman, Adithyavairavan Murali, Clemens Eppner, Bryan Peele, Byron Boots, and Dieter Fox. Motion policy networks. In *Conference on Robot Learning*, pages 967–977. PMLR, 2023. 124, 127, 128, 131, 132, 174, 175, 176, 178, 182

[109] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *ICML*, 2018. 10, 23, 143

[110] Carlos Florensa, Jonas Degrave, Nicolas Heess, Jost Tobias Springenberg, and Martin Riedmiller. Self-supervised learning of image embedding for continuous control. *arXiv preprint arXiv:1901.00943*, 2019. 23

[111] David F Fouhey, Vincent Delaitre, Abhinav Gupta, Alexei A Efros, Ivan Laptev, and Josef Sivic. People watching: Human actions as a cue for single view geometry. In *ECCV*, 2012. 40

[112] Cinzia Freschi, Vincenzo Ferrari, Franca Melfi, Mauro Ferrari, Franco Mosca, and Alfred Cuschieri. Technical review of the da vinci surgical telemanipulator. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 9(4):396–406, 2013. 112

[113] 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. 86, 87

[114] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020. 47, 152

[115] Zipeng Fu, Xuxin Cheng, and Deepak Pathak. Deep whole-body control: learning a unified policy for manipulation and locomotion. In *Conference on Robot Learning*, pages 138–149. PMLR, 2023. 99, 167

[116] Zipeng Fu, Tony Z Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024. 87, 97, 99, 103, 112, 113, 116

[117] Antonino Furnari and Giovanni Maria Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention. In *ICCV*, 2019. 41

[118] Antonino Furnari and Giovanni Maria Farinella. Rolling-unrolling lstms for action anticipation from first-person video. *TPAMI*, 2020. 41

[119] Antonino Furnari, Sebastiano Battiato, Kristen Grauman, and Giovanni Maria Farinella. Next-active-object prediction from egocentric videos. *Journal of Visual Communication and Image Representation*, 2017. 40

[120] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3067–3074, 2015. doi: 10.1109/ICRA.2015.7139620. 122

[121] Jiyang Gao, Zhenheng Yang, and Ram Nevatia. Red: Reinforced encoder-decoder networks for action anticipation. *arXiv preprint arXiv:1707.04818*, 2017. 41

[122] Quentin Garrido, Mahmoud Assran, Nicolas Ballas, Adrien Bardes, Laurent Najman, and Yann LeCun. Learning and leveraging world models in visual representation learning. *arXiv preprint arXiv:2403.00504*, 2024. 137

[123] Dibya Ghosh, Abhishek Gupta, Justin Fu, Ashwin Reddy, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals without reinforcement learning. *arXiv preprint arXiv:1912.06088*, 2019. 18, 46, 139

[124] James J. Gibson. *The Senses Considered as Perceptual Systems*. Houghton Mifflin, Boston, 1966. 39

[125] James J. Gibson. The ecological approach to visual perception. *Houghton Mifflin Comp*, 1979. 39

[126] Rohit Girdhar and Kristen Grauman. Anticipative video transformer. In *ICCV*, 2021. 41

[127] Ankit Goyal, Arsalan Mousavian, Chris Paxton, Yu-Wei Chao, Brian Okorn, Jia Deng, and Dieter Fox. Ifor: Iterative flow minimization for robotic object rearrangement. In *CVPR*, 2022. 46

[128] Mohit Goyal, Sahil Modi, Rishabh Goyal, and Saurabh Gupta. Human hands as probes for interactive object understanding. In *CVPR*, 2022. VIII, XIV, 39, 40, 45, 47, 48, 49, 53, 57, 147, 150, 151, 152

[129] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *CVPR*, 2022. 40, 41

[130] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016. 23

[131] Jiayuan Gu, Devendra Singh Chaplot, Hao Su, and Jitendra Malik. Multi-Skill Mobile Manipulation for Object Rearrangement. In *ICLR*, 2023. 87

[132] Abhinav Gupta, Scott Satkin, Alexei A Efros, and Martial Hebert. From 3d scene geometry to human workspace. In *CVPR*, 2011. 40

[133] 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. 87

[134] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long horizon tasks via imitation and reinforcement learning. *Conference on Robot Learning (CoRL)*, 2019. 18

[135] 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. 86, 87

[136] 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. 86

[137] Agrim Gupta, Lijun Yu, Kihyuk Sohn, Xiuye Gu, Meera Hahn, Li Fei-Fei, Irfan Essa, Lu Jiang, and José Lezama. Photorealistic video generation with diffusion models. *arXiv preprint arXiv:2312.06662*, 2023. 2, 5

[138] Arjun Gupta, Max E Shepherd, and Saurabh Gupta. Predicting motion plans for articulating everyday objects. *arXiv preprint arXiv:2303.01484*, 2023. 100

[139] Reymundo A Gutierrez, Vivian Chu, Andrea L Thomaz, and Scott Niekum. Incremental task modification via corrective demonstrations. In *ICRA*, 2018. 41

[140] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018. 3, 25, 29, 55, 56, 137

[141] Huy Ha and Shuran Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. In *Conference on Robotic Learning (CoRL)*, 2021. 102

[142] 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. 87

[143] 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. 91

[144] Tuomas Haarnoja, Ben Moran, Guy Lever, Sandy H Huang, Dhruva Tirumala, Jan Humplik, Markus Wulfmeier, Saran Tunyasuvunakool, Noah Y Siegel, Roland Hafner, et al. Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *Science Robotics*, 9(89):eadi8022, 2024. 1, 85, 122, 137

[145] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *ICML*, 2019. 3, 12, 27, 29, 33, 55, 56, 137

[146] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *ICLR*, 2020. 3, 14, 25, 27, 29, 55, 56

[147] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *ICLR*, 2021. VIII, 29, 55, 56, 59, 61, 63, 139, 154, 155

[148] Siddhant Haldar, Vaibhav Mathur, Denis Yarats, and Lerrel Pinto. Watch and match: Supercharging imitation with regularized optimal transport. *CoRL*, 2022. 99

[149] Siddhant Haldar, Jyothish Pari, Anant Rai, and Lerrel Pinto. Teach a robot to fish: Versatile imitation from one minute of demonstrations. *arXiv preprint arXiv:2303.01497*, 2023. 99

[150] W. Han, S. Levine, and P. Abbeel. Learning Compound Multi-Step Controllers under Unknown Dynamics. In *IROS*, 2015. 86

[151] Ankur Handa, Karl Van Wyk, Wei Yang, Jacky Liang, Yu-Wei Chao, Qian Wan, Stan Birchfield, Nathan Ratliff, and Dieter Fox. Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9164–9170. IEEE, 2020. 112, 114

[152] Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, Yashraj Narang, Jean-Francois Lafleche, Dieter Fox, and Gavriel State. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. *arXiv*, 2022. 1, 137

[153] Nicklas Hansen, Yixin Lin, Hao Su, Xiaolong Wang, Vikash Kumar, and Aravind Rajeswaran. Modem: Accelerating visual model-based reinforcement learning with demonstrations. *arXiv preprint arXiv:2212.05698*, 2022. 55

[154] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. 7, 122

[155] Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. *ICLR*, 2020. 15, 16, 17, 23, 33

[156] M Hassanin, S Khan, and M Tahtali. Visual affordance and function understanding: a survey. arxiv. *arXiv preprint arXiv:1807.06775*, 2018. 40, 42

[157] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 44, 62, 105, 154, 165

[158] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 30, 33, 148, 152

[159] Alexander Herzog, Kanishka Rao, Karol Hausman, Yao Lu, Paul Wohlhart, Mengyuan Yan, Jessica Lin, Montserrat Gonzalez Arenas, Ted Xiao, Daniel Kappler, et al. Deep rl at scale: Sorting waste in office buildings with a fleet of mobile manipulators. *arXiv preprint arXiv:2305.03270*, 2023. 86, 87, 98, 99

[160] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *CoRR*, abs/2006.11239, 2020. URL `https://arxiv.org/abs/2006.11239`. 69

[161] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models, 2022. 69

[162] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models, 2022. 69

[163] Daniel Honerkamp, Tim Welschehold, and Abhinav Valada. N ˆ{2} m ˆ{2}: Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments. *IEEE Transactions on Robotics*, 39(5):3601–3619, 2023. 87

[164] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. 75

[165] Jiaheng Hu, Peter Stone, and Roberto Martín-Martín. Causal policy gradient for whole-body mobile manipulation. *arXiv preprint arXiv:2305.04866*, 2023. 99

[166] De-An Huang and Kris M Kitani. Action-reaction: Forecasting the dynamics of human interaction. In *ECCV*, 2014. 41

[167] Siyuan Huang, Zan Wang, Puhao Li, Baoxiong Jia, Tengyu Liu, Yixin Zhu, Wei Liang, and Song-Chun Zhu. Diffusion-based generation, optimization, and planning in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16750–16761, 2023. 124

[168] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019. 111

[169] Brian Ichter, Pierre Sermanet, and Corey Lynch. Broadly-exploring, local-policy trees for long-horizon task planning. *arXiv preprint arXiv:2010.06491*, 2020. 124

[170] Advait Jain and Charles C Kemp. Behaviors for robust door opening and doorway traversal with a force-sensing mobile manipulator. Georgia Institute of Technology, 2008. 98, 100

[171] Ashesh Jain, Avi Singh, Hema S Koppula, Shane Soh, and Ashutosh Saxena. Recurrent neural networks for driver activity anticipation via sensory-fusion architecture. In *ICRA*, 2016. 41

[172] Unnat Jain, Svetlana Lazebnik, and Alexander G Schwing. Two can play this game: Visual dialog with discriminative question generation and answering. In *CVPR*, 2018. 41

[173] Unnat Jain, Luca Weihs, Eric Kolve, Mohammad Rastegari, Svetlana Lazebnik, Ali Farhadi, Alexander G. Schwing, and Aniruddha Kembhavi. Two body problem: Collaborative visual task completion. In *CVPR*, 2019. 46

[174] Unnat Jain, Luca Weihs, Eric Kolve, Ali Farhadi, Svetlana Lazebnik, Aniruddha Kembhavi, and Alexander G. Schwing. A cordial sync: Going beyond marginal policies for multi-agent embodied tasks. In *ECCV*, 2020. 41

[175] Stephen James, Kentaro Wada, Tristan Laidlow, and Andrew J Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13739–13748, 2022. 54

[176] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022. 25, 54

[177] 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. 87

[178] Fabian Jenelten, Takahiro Miki, Aravind E Vijayan, Marko Bjelonic, and Marco Hutter. Perceptive locomotion in rough terrain–online foothold optimization. *IEEE Robotics and Automation Letters*, 5(4):5370–5376, 2020. 1

[179] Yunfan Jiang, Chen Wang, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Transic: Sim-to-real policy transfer by learning from online correction. *arXiv preprint arXiv: Arxiv-2405.10315*, 2024. 127

[180] Jacob J Johnson, Linjun Li, Fei Liu, Ahmed H Qureshi, and Michael C Yip. Dynamically constrained motion planning networks for non-holonomic robots. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6937–6943. IEEE, 2020. 124

[181] Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer, 1993. 9, 15, 23

[182] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pages 651–673. PMLR, 2018. 25, 41, 52, 54, 86, 87, 99, 111

[183] 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. *Conference on Robot Learning*, 2021. 54, 86, 87, 99

[184] Aditya Kannan, Kenneth Shaw, Shikhar Bahl, Pragna Mannam, and Deepak Pathak. Deft: Dexterous fine-tuning for real-world hand policies. *CoRL*, 2023. 99

[185] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011. 7, 42, 122

[186] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022. 72

[187] Pushkal Katara, Zhou Xian, and Katerina Fragkiadaki. Gen2sim: Scaling up robot learning in simulation with generative models, 2023. 123

[188] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023. 137

[189] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996. 122

[190] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 77

[191] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986. 122

[192] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024. 112, 122, 123

[193] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 136

[194] Julien Kindle, Fadri Furrer, Tonci Novkovic, Jen Jen Chung, Roland Siegwart, and Juan Nieto. Whole-body control of a mobile manipulator using end-to-end reinforcement learning, 2020. 167

[195] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2014. 12, 145, 166

[196] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 12, 17, 29

[197] 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. 87, 91, 102, 106, 157, 162

[198] Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. 2023. 72, 76, 77

[199] Jens Kober and Jan Peters. Learning motor primitives for robotics. In *ICRA*, 2009. 54

[200] Sven Koenig and Maxim Likhachev. A new principle for incremental heuristic search: Theoretical results. In *ICAPS*, pages 402–405, 2006. 7, 122

[201] Hema S Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. *TPAMI*, 2015. 41

[202] Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. Robot motor skill coordination with em-based reinforcement learning. In *IROS*, 2010. 54

[203] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline Reinforcement Learning with Implicit Q-Learning. In *ICLR*, 2022. 30

[204] 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. 87

[205] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000. 122

[206] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *RSS*, 2021. 1, 54, 97, 111, 122, 123, 137

[207] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *NeurIPS*, 2020. 41

[208] Aviral Kumar, Anikait Singh, Stephen Tian, Chelsea Finn, and Sergey Levine. A workflow for offline model-free robotic reinforcement learning. *Conference on Robot Learning*, 2021. 41

[209] Aviral Kumar, Anikait Singh, Frederik Ebert, Mitsuhiko Nakamoto, Yanlai Yang, Chelsea Finn, and Sergey Levine. Pre-training for robots: Offline rl enables learning new tasks from a handful of trials. *RSS*, 2023. 99

[210] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016. 13

[211] Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024. 68

[212] Tian Lan, Tsung-Chuan Chen, and Silvio Savarese. A hierarchical representation for future action prediction. In *ECCV*, 2014. 41

[213] S. M. Lavalle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and Computational Robotics: New Directions*, 2000. 42

[214] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006. 42

[215] Steven M LaValle and James J Kuffner. Rapidly-exploring random trees: Progress and prospects: Steven m. lavalle, iowa state university, a james j. kuffner, jr., university of tokyo, tokyo, japan. *Algorithmic and computational robotics*, pages 303–307, 2001. 7, 122

[216] Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv:1804.01523*, 2018. 53

[217] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019. 55

[218] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47): eabc5986, 2020. 122

[219] Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. Aligning text-to-image models using human feedback, 2023. 5, 68, 69, 70

[220] Sergey Levine and Vladlen Koltun. Guided policy search. In *ICML*, 2013. 27, 99

[221] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 2016. 25, 27, 41, 45, 52, 99, 111

[222] 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. 87

[223] Yin Li, Miao Liu, and James M Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. In *ECCV*, 2018. 40

[224] Jacky Liang, Jeffrey Mahler, Michael Laskey, Pusong Li, and Ken Goldberg. Using dvrk teleoperation to facilitate deep learning of automation tasks for an industrial robot. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 1–8, 2017. doi: 10.1109/COASE.2017.8256067. 112

[225] Jacky Liang, Saumya Saxena, and Oliver Kroemer. Learning active task-oriented exploration policies for bridging the sim-to-real gap. *arXiv preprint arXiv:2006.01952*, 2020. 99

[226] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. Ara*: Anytime a* with provable bounds on sub-optimality. *Advances in neural information processing systems*, 16, 2003. 7, 122

[227] Toru Lin, Yu Zhang, Qiyang Li, Haozhi Qi, Brent Yi, Sergey Levine, and Jitendra Malik. Learning visuotactile skills with two multifingered hands. *arXiv:2404.16823*, 2024. 112, 113

[228] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 158

[229] Yixin Lin, Austin S. Wang, Giovanni Sutanto, Akshara Rai, and Franziska Meier. Polymetis. `https://facebookresearch.github.io/fairo/polymetis/`, 2021. 152, 155

[230] Dennis V Lindley et al. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956. 10

[231] 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. 86, 87

[232] Iou-Jen Liu, Unnat Jain, Raymond A Yeh, and Alexander Schwing. Cooperative exploration for multi-agent deep reinforcement learning. In *ICML*, 2021. 46

[233] Miao Liu, Siyu Tang, Yin Li, and James M Rehg. Forecasting human-object interaction: joint prediction of motor attention and actions in first person video. In *ECCV*, 2020. 41

[234] Peiqi Liu, Yaswanth Orru, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Ok-robot: What really matters in integrating open-knowledge models for robotics. *arXiv preprint arXiv:2401.12202*, 2024. 98

[235] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9298–9309, 2023. 69

[236] Shaowei Liu, Subarna Tripathi, Somdeb Majumdar, and Xiaolong Wang. Joint hand motion and interaction hotspots prediction from egocentric videos. In *CVPR*, 2022. VIII, 41, 42, 45, 48, 53, 57, 145, 151, 152, 153, 154

[237] 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. 157

[238] Yunze Liu, Yun Liu, Che Jiang, Kangbo Lyu, Weikang Wan, Hao Shen, Boqiang Liang, Zhoujie Fu, He Wang, and Li Yi. Hoi4d: A 4d egocentric dataset for category-level human-object interaction. In *CVPR*, 2022. 40

[239] Dylan P Losey, Andrea Bajcsy, Marcia K O'Malley, and Anca D Dragan. Physical interaction as communication: Learning robot objectives online from human corrections. *IJRR*, 2022. 41

[240] Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018. 27

[241] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *ICLR*, 2023. 4, 39

[242] 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. 87

[243] Guilherme J Maeda, Gerhard Neumann, Marco Ewerton, Rudolf Lioutikov, Oliver Kroemer, and Jan Peters. Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks. *Autonomous Robots*, 2017. 41

[244] Priyanka Mandikal and Kristen Grauman. Dexvip: Learning dexterous grasping with human hand pose priors from video. In *Conference on Robot Learning*, pages 651–661. PMLR, 2022. 55

[245] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What

matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021. 41, 112

[246] Pragna Mannam, Kenneth Shaw, Dominik Bauer, Jean Oh, Deepak Pathak, and Nancy Pollard. Designing anthropomorphic soft hands through interaction. In *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, pages 1–8, 2023. doi: 10.1109/Humanoids57100.2023.10375195. 6, 111, 117

[247] Manus. `https://www.manus-meta.com/`. VR haptic gloves. 113, 114, 171

[248] Roberto Martin-Martin, Michelle A. Lee, Rachel Gardner, Silvio Savarese, Jeannette Bohg, and Animesh Garg. Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks. *IROS*, 2019. 54

[249] Esteve Valls Mascaro, Hyemin Ahn, and Dongheui Lee. Intention-conditioned long-term human egocentric action forecasting@ ego4d challenge 2022. *arXiv preprint arXiv:2207.12080*, 2022. 41

[250] Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. Discovering and achieving goals via world models. *NeurIPS*, 2021. 7, 25, 27, 29, 33, 35, 135

[251] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. ALAN: Autonomously Exploring Robotic Agents in the Real World. In *ICRA*, 2023. 7, 46, 87, 99, 135

[252] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Structured world models from human videos. *RSS*, 2023. 7, 99, 135

[253] Russell Mendonca, Emmanuel Panov, Bernadette Bucher, Jiuguang Wang, and Deepak Pathak. Continuously improving mobile manipulation with autonomous real-world rl. *Conference on Robot Learning (CoRL)*, 2024. 7, 135

[254] Lina Mezghan, Sainbayar Sukhbaatar, Thibaut Lavril, Oleksandr Maksymets, Dhruv Batra, Piotr Bojanowski, and Karteek Alahari. Memory-augmented reinforcement learning for image-goal navigation. In *IROS*, 2022. 46

[255] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 133

[256] Himangi Mittal, Pedro Morgado, Unnat Jain, and Abhinav Gupta. Learning state-aware visual representations from audible interactions. In *NeurIPS*, 2022. 41

[257] Mayank Mittal, David Hoeller, Farbod Farshidian, Marco Hutter, and Animesh Garg. Articulated object interaction in unknown scenes with whole-body mobile manipulation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1647–1654. IEEE, 2022. 99

[258] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King,

Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. 11, 49, 150, 152

[259] Anahita Mohseni-Kabir, Charles Rich, Sonia Chernova, Candace L Sidner, and Daniel Miller. Interactive hierarchical task learning from a single demonstration. In *International Conference on Human-Robot Interaction*, 2015. 41

[260] Movella. Xsens - motion tracking. `https://www.movella.com/products/xsens#overview`. Motion capture technology. 112

[261] Katharina Mülling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013. 54

[262] Adithyavairavan Murali, Arsalan Mousavian, Clemens Eppner, Adam Fishman, and Dieter Fox. Cabinet: Scaling neural collision detection for object rearrangement with procedural scene generation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1866–1874. IEEE, 2023. 133

[263] Fabio Muratore, Christian Eilers, Michael Gienger, and Jan Peters. Data-efficient domain randomization with bayesian optimization. *IEEE Robotics and Automation Letters*, 6(2):911–918, 2021. 99

[264] Austin Myers, Ching L Teo, Cornelia Fermüller, and Yiannis Aloimonos. Affordance detection of tool parts from geometric features. In *ICRA*), 2015. 40

[265] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *arXiv preprint arXiv:1708.02596*, 2017. 55

[266] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep Dynamics Models for Learning Dexterous Manipulation. In *CoRL*, pages 1101–1112. PMLR, 2020. 55, 86, 87

[267] Tushar Nagarajan, Christoph Feichtenhofer, and Kristen Grauman. Grounded human-object interaction hotspots from video. In *ICCV*, 2019. 39, 40, 42, 48, 49, 53, 57, 147, 151, 152

[268] Tushar Nagarajan, Yanghao Li, Christoph Feichtenhofer, and Kristen Grauman. Ego-topo: Environment affordances from egocentric video. In *CVPR*, 2020. 40

[269] K. Nagatani and S.I. Yuta. An experiment on opening-door-behavior by an autonomous mobile robot with a manipulator. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 2, pages 45–50 vol.2, 1995. doi: 10.1109/IROS.1995.526137. 98, 100

[270] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. *ICRA*, 2017. 46

[271] Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *NeurIPS*, pages 9191–9200, 2018. 10, 21, 23, 27, 46, 143

[272] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020. 30

[273] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022. XVI, 4, 32, 35, 39, 41, 48, 49, 51, 53, 55, 60, 62, 63, 145, 151, 152, 154, 155

[274] Soroush Nasiriany, Huihan Liu, and Yuke Zhu. Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2022. 41, 102, 166

[275] OpenAI, Matthias Plappert, Raul Sampedro, Tao Xu, Ilge Akkaya, Vineet Kosaraju, Peter Welinder, Ruben D'Sa, Arthur Petron, Henrique Ponde de Oliveira Pinto, et al. Asymmetric self-play for automatic goal discovery in robotic manipulation. *arXiv preprint arXiv:2101.04882*, 2021. 23, 140

[276] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *NeurIPS*, pages 8617–8629, 2018. 27

[277] Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *ICML*, 2017. 27, 46

[278] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023. 112, 122, 123

[279] Jyothish Pari, Nur Muhammad, Sridhar Pandian Arunachalam, Lerrel Pinto, et al. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021. 32, 41, 48, 107

[280] Simone Parisi, Victoria Dean, Deepak Pathak, and Abhinav Gupta. Interesting object, curious agent: Learning task-agnostic exploration. *NeurIPS*, 2021. 46

[281] Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta. The unsurprising effectiveness of pre-trained vision models for control. *ICML*, 2022. 41

[282] Younghyo Park and Pulkit Agrawal. Using apple vision pro to train and control robots, 2024. URL `https://github.com/Improbable-AI/VisionProTeleop`. 111, 112, 116, 171

[283] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *ICRA*, 2009. 25

[284] Peter Pastor, Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, and Stefan Schaal. Skill learning and task outcome prediction for manipulation. In *ICRA*, 2011. 54

[285] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017. 3, 13, 27, 28, 46, 140

[286] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A Efros, and Trevor Darrell. Zero-shot visual imitation. In *ICLR*, 2018. 23, 46

[287] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International Conference on Machine Learning*, pages 5062–5071, 2019. 3, 13, 25, 27, 28, 29, 31

[288] Georgios Pavlakos, Dandan Shan, Ilija Radosavovic, Angjoo Kanazawa, David Fouhey, and Jitendra Malik. Reconstructing hands in 3D with transformers. In *CVPR*, 2024. 113, 116

[289] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019. 30, 35, 62, 154

[290] Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning (CoRL)*, 2020. 99

[291] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *ICML*, 2007. 62, 154

[292] Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Reinforcement learning for humanoid robotics. In *International Conference on Humanoid Robots*, 2003. 54

[293] L. Peterson, D. Austin, and D. Kragic. High-level control of a mobile manipulator for door opening. In *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, volume 3, pages 2333–2338 vol.3, 2000. doi: 10.1109/IROS.2000.895316. 98, 100

[294] Lerrel Pinto and Abhinav 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. 25, 41, 52

[295] Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *ICML*, 2019. XIII, XVI, XVII, 3, 10, 17, 21, 23, 27, 33, 46, 87, 99, 140, 143

[296] Pascal Poupart, Nikos Vlassis, Jesse Hoey, and Kevin Regan. An analytic solution to discrete bayesian reinforcement learning. In *ICML*, 2006. 27

[297] Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning text-to-image diffusion models with reward backpropagation. *arXiv preprint arXiv:2310.03739*, 2023. 68, 69, 70, 72, 75

[298] Mihir Prabhudesai, Russell Mendonca, Zheyang Qin, Katerina Fragkiadaki, and Deepak Pathak. Video diffusion alignment via reward gradients. *arXiv preprint arXiv:2407.08737*, 2024. 7

[299] M. Prada, A. Remazeilles, A. Koene, and S. Endo. Dynamic movement primitives for human-robot interaction: Comparison with human behavioral observation. In *International Conference on Intelligent Robots and Systems*, 2013. 41

[300] Psyonic. `https://www.psyonic.io`. Bionic hand. 112

[301] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 127

[302] Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. In *ECCV*, 2022. 41, 55

[303] Yuzhe Qin, Binghao Huang, Zhao-Heng Yin, Hao Su, and Xiaolong Wang. Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation. In *Conference on Robot Learning*, pages 594–605. PMLR, 2023. 100

[304] Yuzhe Qin, Wei Yang, Binghao Huang, Karl Van Wyk, Hao Su, Xiaolong Wang, Yu-Wei Chao, and Dietor Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. *RSS*, 2023. 112

[305] Sean Quinlan and Oussama Khatib. Elastic bands: Connecting path planning and control. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 802–807. IEEE, 1993. 122

[306] Ahmed H Qureshi, Anthony Simeonov, Mayur J Bency, and Michael C Yip. Motion planning networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2118–2124. IEEE, 2019. 124, 132

[307] Ahmed H Qureshi, Jiangeng Dong, Austin Choe, and Michael C Yip. Neural manipulation planning on constraint manifolds. *IEEE Robotics and Automation Letters*, 5(4): 6089–6096, 2020. 124

[308] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 53, 68, 72, 104, 168

[309] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. *CoRL*, 2022. 4, 39, 55

[310] Ilija Radosavovic, Bike Zhang, Baifeng Shi, Jathushan Rajasegaran, Sarthak Kamat, Trevor Darrell, Koushil Sreenath, and Jitendra Malik. Humanoid locomotion as next token prediction. *arXiv preprint arXiv:2402.19469*, 2024. 137

[311] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *L4DC*, 2021. 41

[312] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024. 5, 68, 70, 76

[313] Santhosh K Ramakrishnan, Dinesh Jayaraman, and Kristen Grauman. An exploration of embodied visual exploration. *arXiv preprint arXiv:2001.02192*, 2020. 46

[314] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021. 1, 136

[315] Fabio Ramos, Rafael Carvalhaes Possas, and Dieter Fox. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators. *arXiv preprint arXiv:1906.01728*, 2019. 99

[316] Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. Sayplan: Grounding large language models using 3d scene graphs for scalable task planning. *arXiv preprint arXiv:2307.06135*, 2023. 167

[317] Nathan Ratliff, J Andrew Bagnell, and Siddhartha S Srinivasa. Imitation learning for locomotion and manipulation. In *International Conference on Humanoid Robots*, 2007. 25

[318] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE international conference on robotics and automation*, pages 489–494. IEEE, 2009. 7, 122

[319] Nathan Ratliff, Jan Issac, Daniel Kappler, Stan Birchfield, and Dieter Fox. Riemannian motion policies. *arXiv preprint arXiv:1801.02854*, 2018. 41

[320] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems*, 3:297–330, 2020. 112

[321] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022. 55

[322] Allen Z Ren, Hongkai Dai, Benjamin Burchfiel, and Anirudha Majumdar. Adaptsim: Task-driven simulation adaptation for sim-to-real transfer. *arXiv preprint arXiv:2302.04903*, 2023. 99

[323] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014. 12, 17, 29

[324] Nicholas Rhinehart and Kris M Kitani. Learning action maps of large environments via first-person vision. In *CVPR*, 2016. 41

[325] Daniel Ritchie, Kai Wang, and Yu-an Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6182–6190, 2019. 123

[326] Rokoko. `https://www.rokoko.com/`. Motion capture hardware and software. 112, 113

[327] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 1

[328] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *arXiv preprint arXiv:2201.02610*, 2022. 112

[329] Yu Rong, Takaaki Shiratori, and Hanbyul Joo. Frankmocap: A monocular 3d whole-body pose estimation system via regression and integration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1749–1759, 2021. 41, 112, 113

[330] Anirban Roy and Sinisa Todorovic. A multi-scale cnn for affordance segmentation in rgb images. In *ECCV*, 2016. 40

[331] Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1:127–190, 1999. 60

[332] Reuven Y Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997. 31

[333] R.B. Rusu, W. Meeussen, Sachin Chitta, and Michael Beetz. Laser-based perception for door and handle identification. pages 1 – 8, 07 2009. 100

[334] Edward L. Ryan, Richard; Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary Educational Psychology*, 2000. 2

[335] Kallol Saha, Vishal Mandadi, Jayaram Reddy, Ajit Srikanth, Aditya Agarwal, Bipasha Sen, Arun Singh, and Madhava Krishna. Edmp: Ensemble-of-costs-guided diffusion for motion planning. *arXiv preprint arXiv:2309.11414*, 2023. 124, 132

[336] Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation. *arXiv preprint arXiv:2403.12015*, 2024. 69

[337] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8), 1964. 145

[338] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347, 2019. 99

[339] Johann Sawatzky, Abhilash Srikantha, and Juergen Gall. Weakly supervised affordance detection. In *CVPR*, 2017. 40

[340] Vaibhav Saxena, Jimmy Ba, and Danijar Hafner. Clockwork variational autoencoders. *arXiv preprint arXiv:2102.09532*, 2021. 12

[341] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320, 2015. 9, 23

[342] C. Schenck and D. Fox. Visual Closed-Loop Control for Pouring Liquids. In *ICRA*, 2017. 86, 87

[343] Jürgen Schmidhuber. Curious model-building control systems. In *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pages 1458–1463. IEEE, 1991. 13, 29, 56

[344] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers, 1991. 29, 56

[345] C Schuhmann. Laoin aesthetic predictor. 2022. URL `https://laion.ai/blog/laion-aesthetics/`. 68, 69, 73, 76, 77

[346] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014. 7, 122

[347] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 70, 76

[348] Laura Schulz. Finding new facts; thinking new thoughts. *Advances in child development and behavior*, 43:269–294, 2012. 10

[349] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. *ICML*, 2020. XIV, XVI, XVII, 10, 13, 14, 17, 21, 22, 25, 27, 29, 31, 140, 144

[350] Pierre Sermanet, Kelvin Xu, and Sergey Levine. Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*, 2016. 57

[351] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from pixels. 2017. 41

[352] Shadow Robot Company. `https://www.shadowrobot.com/`. Robotic hand. 112, 117

[353] 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. 87, 97

[354] Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. Vint: A foundation model for visual navigation, 2023. 97

[355] Rutav M Shah and Vikash Kumar. Rrl: Resnet as representation for reinforcement learning. In *ICML*, 2021. 4, 39, 41

[356] Dandan Shan, Jiaqi Geng, Michelle Shu, and David Fouhey. Understanding human hands in contact at internet scale. In *CVPR*, 2020. 39, 41, 43, 53, 57, 62, 145, 152, 153, 155

[357] Lin Shao, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. *The International Journal of Robotics Research*, 40(12-14):1419–1434, 2021. 27, 41, 57

[358] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019. 27

[359] 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. 87

[360] Pratyusha Sharma, Deepak Pathak, and Abhinav Gupta. Third-person visual imitation learning via decoupled hierarchical controller. *arXiv preprint arXiv:1911.09676*, 2019. 41, 45, 57

[361] Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. Videodex: Learning dexterity from internet videos. In *CoRL*, 2022. 41, 55, 57

[362] Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *arXiv preprint arXiv:2309.06440*, 2023. XII, XVII, 6, 111, 115, 117, 118, 119

[363] Kenneth Shaw, Yulong Li, Jiahui Yang, Mohan Srirama, Ray Liu, Haoyu Xiong, Russell Mendonca, and Deepak Pathak. Bimanual dexterity for complex tasks. *Conference on Robot Learning (CoRL)*, 2024. 7

[364] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *CoRL*, 2022. 46, 54

[365] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023. 54, 123

[366] Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. *arXiv preprint arXiv:1810.12162*, 2018. 10, 13

[367] Paul J Silvia. Curiosity and motivation. *The Oxford handbook of human motivation*, pages 157–166, 2012. 2

[368] Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-End Robotic Reinforcement Learning without Reward Engineering. In *RSS*, 2019. 86, 87

[369] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube. *RSS*, 2022. 41, 112, 114

[370] Laura Smith, Nikita Dhawan, Marvin Zhang, Pieter Abbeel, and Sergey Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. *RSS*, 2020. 41, 57, 86, 87

[371] 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. 86, 87, 99

[372] 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. 87

[373] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015. 69

[374] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. 72

[375] Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5(3):4978–4985, 2020. 41, 113

[376] Jordi Spranger, Roxana Buzatoiu, Athanasios Polydoros, Lazaros Nalpantidis, and Evangelos Boukas. Human-machine interface for remote training of robot tasks. *arXiv preprint arXiv:1809.09558*, 2018. 41

[377] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, Karen Liu, et al. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on Robot Learning*, pages 477–490. PMLR, 2022. 98, 99

[378] 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. 156

[379] A. Strehl and M. Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 2008. 27

[380] Marlin P Strub and Jonathan D Gammell. Adaptively informed trees (ait*): Fast asymptotically optimal path planning through adaptive heuristics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3191–3198. IEEE, 2020. 7, 122, 126, 131, 174, 175

[381] F. Stulp, E. A. Theodorou, and S. Schaal. Reinforcement learning with sequences of motion primitives for robust manipulation. *Transactions on Robotics*, 2012. 54

[382] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *ICLR*, 2018. 23

[383] 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. 86, 87, 98

[384] Yi Sun, Faustino Gomez, and Jürgen Schmidhuber. Planning to be surprised: Optimal bayesian exploration in dynamic environments. In *International Conference on Artificial General Intelligence*, pages 41–51. Springer, 2011. 13

[385] Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, et al. Curobo: Parallelized collision-free robot motion generation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8112–8119. IEEE, 2023. 130, 131

[386] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 11

[387] Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 1999. 54

[388] Michita Imai Takuma Seno. d3rlpy: An offline deep reinforcement library. In *NeurIPS 2021 Offline Reinforcement Learning Workshop*, December 2021. 150, 152

[389] Corentin Tallec and Yann Ollivier. Unbiasing truncated backpropagation through time. *arXiv preprint arXiv:1705.08209*, 2017. 72

[390] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. *NeurIPS*, 2017. 27, 46

[391] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018. XIV, 18, 21, 144

[392] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 135, 136

[393] Stephen Tian, Suraj Nair, Frederik Ebert, Sudeep Dasari, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Model-based visual planning with self-supervised functional distances. *arXiv preprint arXiv:2012.15373*, 2020. 23

[394] 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. 85

[395] Josh Tobin, Lukas Biewald, Rocky Duan, Marcin Andrychowicz, Ankur Handa, Vikash Kumar, Bob McGrew, Alex Ray, Jonas Schneider, Peter Welinder, et al. Domain randomization and generative models for robotic grasping. In *IROS*, 2018. 54

[396] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022. 68, 73, 76, 77

[397] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 1

[398] Shagun Uppal, Ananye Agarwal, Haoyu Xiong, Kenneth Shaw, and Deepak Pathak. Spin: Simultaneous perception, interaction and navigation. *arXiv preprint arXiv:2405.07991*, 2024. 123

[399] Yusuke Urakami, Alec Hodgkinson, Casey Carlin, Randall Leu, Luca Rigazio, and Pieter Abbeel. Doorgym: A scalable door opening environment and baseline agent. *arXiv preprint arXiv:1908.01887*, 2019. 100

[400] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *NeurIPS*, 33:19667–19679, 2020. 33, 63, 154

[401] Valve Corporation. `https://store.steampowered.com/steamvr`. Virtual reality platform. 111, 112

[402] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 44, 154

[403] Vicon. `https://www.vicon.com/`. Motion capture technology. 111

[404] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. *arXiv preprint arXiv:1706.08033*, 2017. 41

[405] Carl Vondrick, Deniz Oktay, Hamed Pirsiavash, and Antonio Torralba. Predicting motivations of actions by leveraging text. In *CVPR*, 2016. 41

[406] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023. 112

[407] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. *arXiv preprint arXiv:2311.12908*, 2023. 70, 76

[408] Chen Wang, Haochen Shi, Weizhuo Wang, Ruohan Zhang, Li Fei-Fei, and C Karen Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *arXiv preprint arXiv:2403.07788*, 2024. 6, 111, 112, 114

[409] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. 135

[410] Jiayu Wang, Shize Lin, Chuxiong Hu, Yu Zhu, and Limin Zhu. Learning semantic keypoint representations for door opening manipulation. *IEEE Robotics and Automation Letters*, 5(4):6980–6987, 2020. doi: 10.1109/LRA.2020.3026963. 100

[411] Jiuniu Wang, Hangjie Yuan, Dayou Chen, Yingya Zhang, Xiang Wang, and Shiwei Zhang. Modelscope text-to-video technical report. *arXiv preprint arXiv:2308.06571*, 2023. 70, 76

[412] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. In *2021 International Conference on 3D Vision (3DV)*, pages 106–115. IEEE, 2021. 123

[413] Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023. 123

[414] David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*, 2018. XIII, XVI, XVII, 10, 17, 21, 23, 27, 143, 144

[415] Charles W Warren. Global path planning using artificial potential fields. In *1989 IEEE International Conference on Robotics and Automation*, pages 316–317. IEEE Computer Society, 1989. 122

[416] Justin Wasserman, Karmesh Yadav, Girish Chowdhary, Abhinav Gupta, and Unnat Jain. Last-mile embodied visual navigation. In *CoRL*, 2022. 46

[417] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *neural information processing systems*, 2015. 12, 23

[418] Théophane Weber, Sébastien Racanière, David P Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. *arXiv preprint arXiv:1707.06203*, 2017. 55

[419] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992. 104

[420] Josiah Wong, Albert Tung, Andrey Kurenkov, Ajay Mandlekar, Li Fei-Fei, Silvio Savarese, and Roberto Martín-Martín. Error-aware imitation learning from teleoperation data for mobile manipulation. In *Conference on Robot Learning*, pages 1367–1378. PMLR, 2022. 99

[421] Bohan Wu, Suraj Nair, Li Fei-Fei, and Chelsea Finn. Example-driven model-based reinforcement learning for solving long-horizon visuomotor tasks. *arXiv preprint arXiv:2109.10312*, 2021. 53

[422] Bohan Wu, Roberto Martin-Martin, and Li Fei-Fei. M-EMBER: Tackling Long-Horizon Mobile Manipulation via Factorized Domain Transfer. In *ICRA*, 2023. 87, 99

[423] Chengzhi Wu, Xuelei Bi, Julius Pfrommer, Alexander Cebulla, Simon Mangold, and Jürgen Beyerer. Sim2real transfer learning for point cloud segmentation: An industrial application case on autonomous disassembly. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4531–4540, 2023. 127

[424] 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. 87, 99

[425] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and Pieter Abbeel. DayDreamer: World Models for Physical Robot Learning. In *CoRL*, 2023. 55, 86, 87

[426] Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. *arXiv preprint arXiv:2309.13037*, 2023. 6, 111, 112, 113, 115, 171

[427] Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023. 68, 69, 72, 76, 77

[428] 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. 87

[429] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pretraining for motor control. *Conference on Robot Learning*, 2022. 4, 39, 41, 53, 55

[430] Haoyu Xiong, Quanzhou Li, Yun-Chun Chen, Homanga Bharadhwaj, Samarth Sinha, and Animesh Garg. Learning by watching: Physical imitation of manipulation skills from human videos. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7827–7834, 2021. doi: 10.1109/IROS51168.2021.9636080. 57, 99, 103

[431] Haoyu Xiong, Haoyuan Fu, Jieyi Zhang, Chen Bao, Qiang Zhang, Yongxi Huang, Wenqiang Xu, Animesh Garg, and Cewu Lu. Robotube: Learning household manipulation from human videos with simulated twin environments. In *6th Annual Conference on Robot Learning*, 2022. URL `https://openreview.net/forum?id=VD0nXUG5Qk`. 99, 103

[432] 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. 7, 135

[433] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation, 2023. 68, 69

[434] 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. 86, 87

[435] Mengda Xu, Manuela Veloso, and Shuran Song. Aspire: Adaptive skill priors for reinforcement learning. *Advances in Neural Information Processing Systems*, 35:38600–38613, 2022. 99

[436] Karmesh Yadav, Ram Ramrakhya, Arjun Majumdar, Vincent-Pierre Berges, Sachit Kuhar, Dhruv Batra, Alexei Baevski, and Oleksandr Maksymets. Offline visual representation learning for embodied navigation. *arXiv:2204.13226*, 2022. 4, 39, 41

[437] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine. Collective Robot Reinforcement Learning with Distributed Asynchronous Guided Policy Search. In *IROS*, 2017. 86, 87

[438] Mengyuan Yan, Gen Li, Yilin Zhu, and Jeannette Bohg. Learning topological motion primitives for knot planning. In *IROS*, 2020. 41

[439] Ruihan Yang, Yejin Kim, Aniruddha Kembhavi, Xiaolong Wang, and Kiana Ehsani. Harmonic Mobile Manipulation. *arXiv preprint arXiv:2312.06639*, 2023. 85, 97, 167

[440] Denis Yarats, Rob Fergus, and Ilya Kostrikov. Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels. In *ICLR*, 2021. 91

[441] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021. XIV, 22, 144

[442] Jianglong Ye, Jiashun Wang, Binghao Huang, Yuzhe Qin, and Xiaolong Wang. Learning continuous grasping function with a dexterous hand from human demonstrations. *arXiv preprint arXiv:2207.05053*, 2022. 41

[443] Yufei Ye, Abhinav Gupta, and Shubham Tulsiani. What's in your hands? 3d reconstruction of generic objects in hands. In *CVPR*, 2022. 41

[444] Sriram Yenamandra, Arun Ramachandran, Karmesh Yadav, Austin Wang, Mukul Khanna, Theophile Gervet, Tsung-Yen Yang, Vidhi Jain, Alexander William Clegg, John Turner, et al. Homerobot: Open-vocabulary mobile manipulation. *arXiv preprint arXiv:2306.11565*, 2023. 98, 99

[445] 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. 87, 99

[446] Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. Visual imitation made easy. *arXiv preprint arXiv:2008.04899*, 2020. 41

[447] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018. 45

[448] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR, 2020. 18

[449] 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. 87

[450] Hangjie Yuan, Shiwei Zhang, Xiang Wang, Yujie Wei, Tao Feng, Yining Pan, Yingya Zhang, Ziwei Liu, Samuel Albanie, and Dong Ni. Instructvideo: Instructing video diffusion models with human feedback. *arXiv preprint arXiv:2312.12490*, 2023. 70

[451] Kevin Zakka, Andy Zeng, Pete Florence, Jonathan Tompson, Jeannette Bohg, and Debidatta Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning. *Conference on Robot Learning*, 2021. 57

[452] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. *arXiv:1803.09956*, 2018. 41, 46, 54

[453] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, and Johnny Lee. Transporter networks: Rearranging the visual world for robotic manipulation. *CoRL*, 2020. 41, 54

[454] Kuo-Hao Zeng, Luca Weihs, Ali Farhadi, and Roozbeh Mottaghi. Pushing it out of the way: Interactive visual navigation. In *CVPR*, 2021. 46

[455] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 15

[456] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 5628–5635. IEEE, 2018. 41

[457] Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Automatic curriculum learning through value disagreement. *NeurIPS*, 33, 2020. 10, 23, 27

[458] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *RSS*, 2023. XVII, 1, 111, 112, 113, 114, 119, 171, 177

[459] Yibiao Zhao and Song-Chun Zhu. Scene parsing by integrating function, geometry and appearance models. In *CVPR*, 2013. 40

[460] Yizhou Zhao, Qiaozi Gao, Liang Qiu, Govind Thattai, and Gaurav S Sukhatme. Opend: A benchmark for language-driven door and drawer opening. *arXiv preprint arXiv:2212.05211*, 2022. 99

[461] Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all, March 2024. URL `https://github.com/hpcaitech/Open-Sora`. 75, 76

[462] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IX*, pages 350–368. Springer, 2022. VII, 28, 31, 32, 48, 61, 91, 102, 106, 157, 162

[463] 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. 87

[464] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. *ICRA*, 2017. 46

[465] Yixin Zhu, Chenfanfu Jiang, Yibiao Zhao, Demetri Terzopoulos, and Song-Chun Zhu. Inferring forces and learning human utilities from videos. In *CVPR*, 2016. 40

[466] Yuke Zhu, Alireza Fathi, and Li Fei-Fei. Reasoning about object affordances in a knowledge base representation. In *ECCV*, 2014. 40

[467] Ziwen Zhuang, Zipeng Fu, Jianren Wang, Christopher Atkeson, Soeren Schwertfeger, Chelsea Finn, and Hang Zhao. Robot parkour learning. *arXiv preprint arXiv:2309.05665*, 2023. 122, 123