Dexterous Manipulation from Images: Autonomous Real-World RL via Substep Guidance

Kelvin Xu*[1] Zheyuan Hu*[1] Ria Doshi[1] Aaron Rovinsky[1] Vikash Kumar[2] Abhishek Gupta[3] Sergey Levine[1]
[1] UC Berkeley [2] Meta AI Research [3] University of Washington

Fig. 1: Filmstrip of the final learned brush skill. Our agent is able to learn to grasp, in-hand reorient, and brush a surface using a kitchen brush. After around $\sim$ 18 hours of unattended learning, our system successfully performs each of these sub-tasks with $\geq 80\%$ success.

***Abstract*— Complex and contact-rich robotic manipulation tasks, particularly those that involve multi-fingered hands and underactuated object manipulation, present a significant challenge to any control method. Methods based on reinforcement learning offer an appealing choice for such settings, as they can enable robots to learn to delicately balance contact forces and dexterously reposition objects without strong modeling assumptions. However, running reinforcement learning on real-world dexterous manipulation systems often requires significant manual engineering. This negates the benefits of autonomous data collection and ease of use that reinforcement learning should in principle provide. In this paper, we describe a system for vision-based dexterous manipulation that provides a "programming-free" approach for users to define new tasks and enable robots with complex multi-fingered hands to learn to perform them through interaction. The core principle underlying our system is that, in a vision-based setting, users should be able to provide high-level intermediate supervision that circumvents challenges in teleoperation or kinesthetic teaching which allow a robot to not only learn a task efficiently but also to autonomously practice. Our system includes a framework for users to define a final task and intermediate sub-tasks with image examples, a reinforcement learning procedure that learns the task autonomously without interventions, and experimental results with a four-finger robotic hand learning multi-stage object manipulation tasks directly in the real world, without simulation, manual modeling, or reward engineering.**

I. Introduction

Complex and contact-rich robotic manipulation tasks, particularly those that involve multi-fingered robotic hands and underactuated objects, present significant challenges to any control method. Reinforcement learning (RL) offers an appealing choice for such settings, as it in principle enables a robot to learn to adeptly apply contact forces and manipulate objects without strong modeling assumptions, directly from real-world experience. However, running RL on real-world robotic platforms raises a number of practical issues that lie outside the standard RL formulation, such as difficulties with reward specification, state estimation and the practicalities of autonomous training. Addressing such issues typically requires significant manual engineering or human intervention. This has led researchers to study alternative solutions, such as transfer from simulation [1]–[3], imitation learning [4]–[6], or use of cumbersome instrumentation, such as motion capture [2], [7]. Even when these issues can be overcome, effective real-world reinforcement learning typically requires considerable reward engineering [8], complex reset mechanisms or scripts [9], and other manually designed components. Each of these solutions erodes the original benefits of autonomy and ease of use that RL should in principle provide. Solving even the simplest tasks with RL requires considerable domain and robotics expertise to program reward functions and reset mechanisms for autonomous operation. Thus, in order to allow for learning-based dexterous manipulation systems to reach their full potential in terms of practicality, accessibility, and scalability, it is critical to limit the assumptions on manual engineering while still providing enough supervision for reinforcement learning to be tractable.

In this work, we propose a robotic learning system that can learn to control high-dimensional multi-fingered robotic hands from raw visual observations, without the need for extensive engineering for every new task. In the absence of simulation, manual reward shaping, and hand-designed state estimation instrumentation, we aim to enable RL to be as autonomous as possible. The robot should be able to practice the task for a long period of time without human intervention, and the task itself should be specified in a way that does not require per-task programming or human-in-the-loop supervision. To this end, we propose a system that autonomously practices a sequence of sub-skills based on high-level milestone specifications provided by the user that break up a complex task into more manageable sub-problems. The milestone specifications consist of snapshots of critical states illustrated by posing the robot and objects in the scene. For multi-fingered hands, such examples are significantly easier to provide than full demonstrations, and our system can use them to learn reward functions that provide sufficient shaping for RL in the real-world without per-task engineering or specific reward design. The system uses multi-camera visual observations to localize and manipulate

*Both authors contributed equally
https://sites.google.com/view/dexterous-avail/

objects, with policies learned end-to-end from pixels and no motion capture. By sequencing the sub-skills appropriately and introducing very simple physical instrumentation (in our experiments, tethering the object to prevent it from falling out of reach), the robot can learn dexterous behaviors by practicing for up to 48 hours fully autonomously. The milestone decomposition makes both reward inference and autonomous practicing significantly easier, enabling real-world learning of complex tasks. Our experiments (see Fig. 1 for an example) show that this approach can learn skills that involve basic grasping, in-hand reorientation, and object manipulation, through significant amounts of practicing, entirely from images and without task-specific reward engineering.

II. Related Work

Prior work has studied control of complex hands using trajectory optimization [10], [11], policy search [12]–[14], simulation to real-world transfer [3], [15], [16], and real-world reinforcement learning [17], [18]. In contrast to our work, the majority of this prior work has assumed access to compact state representations or accurate simulators and object models. Closer to the system we describe in this paper is prior work on learning visuomotor policies for dexterous manipulation [19]–[21] However, with the exception of some work we discuss below, prior systems on RL for dexterous manipulation typically require assumptions on manually designed rewards, or ground truth object state observations. These assumptions hinder the application of RL in more real-world settings.

An important consideration in our system is the ability to specify a task without manual reward engineering, by using intermediate milestone examples. Previously studied methods for task specification include having humans provide demonstrations for imitation learning [4], [22], [23], using inverse RL [24]–[26], active settings where users can provide corrections [27]–[29], or ranking-based preferences [30], [31]. While some prior work [32], [33] also uses subgoals, these are firstly restricted to reaching only particular goal states rather than more abstract milestones and are only applied in much simpler simulated problems with perfect state estimation. Motivated by the goal of broader applicability, we do not assume access to expert demonstrations (e.g., via teleoperation or kinesthetic teaching), which can themselves be difficult to provide for high-dimensional systems [34], [35]. For example, providing kinesthetic demonstrations for a full hand-arm robotic system requires very challenging coordination and several simultaneous demonstrators, and is incompatible with vision-based policy learning (as the demonstrator is in the scene, and often occludes the robot or objects). In contrast, we utilize sparse images of intermediate outcomes that can be obtained simply by positioning the robot and object in particular states and build on the VICE framework [36] for reward inference. Our focus is not on devising a new *algorithm* for learning rewards, but on leveraging existing components, such as VICE, to build a complete, autonomous, robotic system that can enable scalable RL with a dexterous manipulator in the real world.

TABLE I: A comparison between the assumptions of AVAIL and prior autonomous RL methods.

Method	No Hand Engineered Reward	Multi-Task	Vision	High-DoFs
R3L [39]	✓	×	✓	×
MTRF [38]	×	✓	×	✓
Ours	✓	✓	✓	✓

Therefore, although some of the building blocks of our system are based on prior work, their combination and the capabilities they enable (learning image-based dexterous manipulation in the real world) are novel.

The most closely related robotic RL systems that have been previously proposed are R3L [37] and MTRF [38]. Our assumptions regarding lightweight instrumentation and vision-based autonomous learning most resemble those of Zhu et al. (2020) [37] (R3L). However, our work tackles a considerably more challenging setting: while Zhu et al. (2020) [37] studied a 3-finger claw mounted on a fixed base, we show that our method can control a 4-fingered hand on a 7 DoF arm. This is done by leveraging a multi-task RL formulation that builds on ideas from MTRF [38] instead of the novelty-based resets in R3L [39], which scale poorly in higher dimensional settings. In contrast to these prior works, our focus is on providing a framework that can enable vision-based learning of object manipulation skills with high-dimensional hands via a lightweight milestone-based task specification mechanism. Part of this requires an automated RL system that can run continuously for 48 hours, though unlike MTRF [38], we still employ lightweight physical instrumentation (by tethering the object to prevent it from falling outside of grasping range). We instead focus on the separate challenges of visual perception and reward specification, avoiding the need for manual reward engineering of the sort used by MTRF and completely circumventing the requirement for motion capture that was crucial in MTRF. We summarize these key system-level differences in Table I.

III. Robotic Platform and Problem Overview

We first present an overview of our robot platform, describing the hardware and task setup, as well as the observation and action space. Then, we provide an overview of our problem setting, focusing on the practical goals of our system. We provide complete details related to our robotic platform in our project website[1] along with details of a simulated analogue that we employ for analysis and ablation experiments.

Our robotic system consists of a custom-built, 4-finger, 16-DoF robot hand, mounted on a 7-DoF Sawyer robotic arm. The arm and hand assembly are positioned over a tabletop surface (Fig. 3, left image). Our policy, which we operate at 8Hz, directly controls each joint position in addition to the Cartesian position and orientation of the arm, resulting in a 22-dimensional action space and 29-dimensional state space. The system is designed to operate for upwards of

[1] https://sites.google.com/view/dexterous-avail/

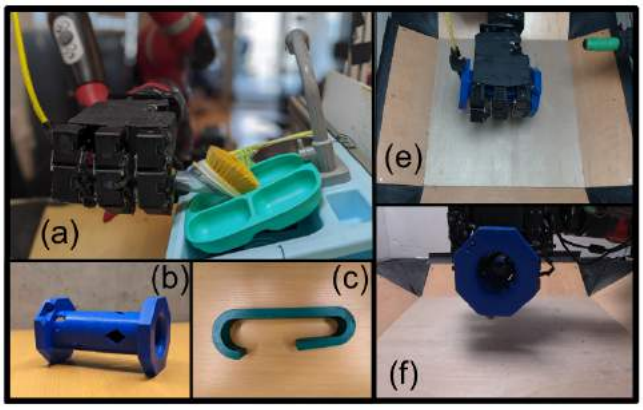

Fig. 3: An overview of our experimental platform: (a) Our robot consists of a 16 DoF four-finger hand mounted on a 7 DoF Sawyer arm; (a, b, c) Objects the robot manipulates in our experiments: a kitchen brush, a cylindrical hose connector, and a hook that must be attached to a handle; (e, f) Observations for the robot come from two monocular RGB cameras.

48 hours in contact-rich environments without breakage. In addition to the robot's own joint encoders, two RGB image observations are provided to the robot via two low-cost web cameras and resized to 84×84. We discuss additional details on our project website.

Our tasks consist of manipulation behaviors such as reaching, grasping, in-hand and mid-air reorienting, and inserting. We consider three tasks (shown in Fig 3) for interacting with several different objects: inserting a hose into a connector on the side of the arena, hooking a rope onto a fixture, and cleaning a surface with a kitchen brush. Successfully completing each of these tasks requires correctly sequencing a series of sub-skills. For example, in order to complete our kitchen task, the robot must grasp and reorient that brush palm down without dropping it before making contact with the surface. Constructing and tuning both manual rewards and state estimation systems for each of these tasks separately would typically require laborious human engineering. For each of these tasks however, the only supervision we assume is to allow the user to place the robot and object in the desired position and capture a set of image "snapshots". We describe how we use this supervision to drive reward inference and task selection via autonomous reinforcement learning in the sections below.

IV. Problem Formalism and User Assumptions

In this section, we formalize our problem setting and supervision assumptions. Consider first the Markov decision process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, p_{dyn}, \rho, \gamma, R)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action spaces, $p_{dyn} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}_{\geq 0}$ denotes the environment dynamics, $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ denotes the reward function, $\rho : \mathcal{S} \mapsto \mathbb{R}_{\geq 0}$ denotes the initial state distribution and $\gamma \in [0, 1)$ denotes the discount factor. The typical objective of episodic RL is to optimize the discounted return $J(\pi) = \mathbb{E}_{\tau \sim \pi}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)]$ with respect to the policy π, where $\tau = \{(s_i, a_i)\}_{i=0}^{T-1}$ is obtained by sampling $s_0 \sim \rho(\cdot)$, $a_t \sim \pi(\cdot \mid s_t)$ and $s_{t+1} \sim p(\cdot \mid s_t, a_t)$.

A principal concern of our work is to ask the question of how best to instantiate RL systems in the real world with minimal per-task engineering, instrumentation and intervention. Standard RL assumes a reward function R that in practice must often be hand engineered and tuned per-task by a user. This challenge is particularly acute in the dexterous manipulation setting where the desired behavior can often itself be composed of a sequence of complex "sub-tasks" (e.g., grasping, re-orienting, etc) with different objects that would need to be instrumented separately. In addition, independent of being challenging to learn, these sub-tasks must be appropriately sequenced in order to complete the task but also to allow the agent to continue to practice in the event of failure. This necessitates the provision of more fine-grained guidance via user supervision, while carefully balancing the cost of providing such supervision. Furthermore, most RL algorithms assume that the environment is episodic and resets are provided for free. This is not true when considering large scale autonomous operation.

To provide fine-grained supervision both for reward inference and autonomous practicing, we propose a method where a user supplies the robot with a set of sub-problems to practice. These sub-problems are defined by "milestone" examples, which constitute a graph structure:

Definition IV.1 (*Milestones graph*)**.** We assume the user provides a set of outcome images that can be summarized by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of cardinality $|K|$ indexed by z, where each vertex $v \in \mathcal{V}$ is composed of a set of M outcome images $\{s_i^z\}_{i=1}^{M}$. Each set of outcome images characterizes a semantically meaningful sub-task to be solved. In addition, upon accomplishing a sub-task, a directed edge $(v, v') \in \mathcal{E}$, or equivalently a binary label, is provided which indicates which sub-task is to be practiced next.

Consistent with the goal of having the agent continuously practice (i.e., not get stuck), we assume there are no sink nodes in the provided graph.[2] Then, instead of optimizing a single-task objective $J(\pi)$, we instead optimize all of the sub-tasks in the milestone graph simultaneously, resulting in a multi-task RL problem. Concretely, we learn a set of K policies π_z indexed by a categorical variable z (one for each milestone), optimizing a set of MDPs, $\mathcal{M} \equiv (\mathcal{S}, \mathcal{A}, p_{dyn}(s_{t+1}|a_t, s_t), \{R_z\}_{z=0}^{K-1}, p_{\text{task}}(z|s))$, where we have introduced a per milestone reward R_z and task predictor p_{task}. This leads to the following objective:

$$J_{\text{MT}}(\{\pi_i\}_{i=0}^{K-1}) = \sum_{i=0}^{\infty} \left[\mathop{\mathbb{E}}_{\substack{s_0^i = s_T^{i-1} \\ \tau \sim \pi_{z_i}}} \left[\sum_{t=0}^{T} \gamma^t R_z(s_t^i, a_t^i) \right] \right] \quad (1)$$

$$z_{i+1} \sim p_{\text{task}}(z_{i+1}|s_T^i). \quad (2)$$

[2]This assumption conceivably could be lifted by providing handling of safety-critical states to avoid irreversible sinks, [40], [41]. For simplicity, we leave addressing safety issues for future work.

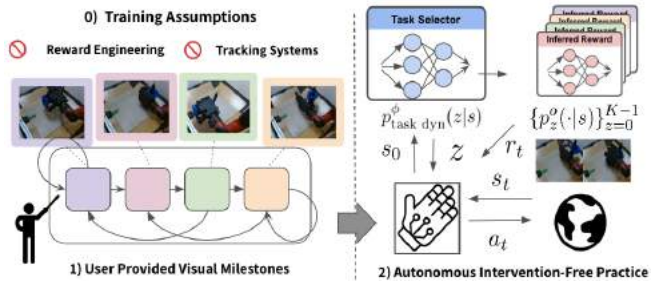

Fig. 4: An overview of our approach. The user provides a set of visual milestones (bottom left, $K = 4$ here) and transitions. We use this to formulate a multi-task learning problem where we leverage this supervision to learn all the components necessary for intervention-free learning. Prior to training, we learn a task selection model (blue module), which is used to choose amongst a set of K policies for data collection (Sec. V-B). This data is used to learn a success classifier (top right, Sec. V-A), which is used to automatically assign rewards. We show that our system is capable of autonomously learning complex manipulation behaviors on a real world anthropomorphic hand with modest instrumentation beyond the robot's own joint encoders and camera.

Formulating the problem in this manner makes conspicuous the need to define and sample from $p_{\text{task}}(z_{i+1}|s^i_T)$ and $R_z(s_t, a_t)$ which can be used to autonomously direct training. To tractably learn these two functions, we leverage the user-provided milestone supervision, which we denote as $D_k = \{s_i\}_{i=1}^N$, and a set of categorical labels y_k from the milestones graph indicating which task should transition to the next. These could, for example, be a set of images showing the robot repositioning and re-grasping the object, and a label indicating the next step is pickup. In the following sections, we show how to learn the rewards R_z, the sub-task transition function $p_{\text{task}}(z_{i+1}|s^i_T)$ and per sub-task policies π_z using the milestone image supervision provided upfront.

V. The AVAIL System: Autonomy via User-Provided Milestones

To address the problem described in Section III, we present AVAIL (Autonomy ViA mILestones) – our system for learning autonomously with minimal intervention and external environment instrumentation. AVAIL (see Fig. 4 for an overview) reframes the RL training process as a multi-task problem that can learn directly from sparse milestone examples provided by users, with minimal external instrumentation or intervention. This makes the process of solving complex tasks with "programming-free" reinforcement learning significantly more approachable. By leveraging the user-provided milestones defined in Sec III, at a high level our system functions (as shown in Fig 4) by (1) deriving reward functions via learned success classifiers, (2) optimizing these rewards in a sample-efficient manner using a multi-task vision-based RL system, and (3) determining which of these tasks to perform given the current robot observations so as to continue practicing autonomously.

A. Visual Multi-Task Policy and Reward Learning from User Milestones

A critical enabler of autonomous learning is the ability for the robot to assign rewards to its own experience. This importantly relieves the burden of manual reward engineering, but comes with the trade-off of removing the ability of the designer to provide task information (e.g., via reward shaping), which can be crucial for tractable learning and directed exploration for long horizon tasks. To resolve this challenge in our setting where we similarly require compound behavior, we leverage the sparse milestone supervision to learn a set of success classifiers $\{p^o_z(\cdot|s)\}_{z=0}^{K-1}$ that decomposes the entire long horizon task. Importantly, the provision of a number of significant milestones takes the burden off a single learned classifier to provide accurate reward shaping.

For each individual classifier $p^o_z(\cdot|s)$, we build on the VICE algorithm [36], extending it to a multi-task setting. We learn a binary classifier $p^o_z(\cdot|s)$ over the set of user-provided examples $\{D_1, D_2, \ldots, D_K\}$ as the positive class, and the agent's own experience sampled on-policy as the negative class. Once trained, the classifier probability $p^o_z(o|s)$, or a monotonic transformation of it (e.g., $\log p^o_z(\cdot|s)$), can be used as R_z from Section III. An added advantage of classifier-based rewards is the property that, in practice, they can often provide some additional degree of shaping [36], [42].

Finally, in order to learn in the real world from raw sensory inputs, a core component of our system is a sample-efficient, multi-task RL algorithm that allows us to learn the robot's joint encoders and raw image observations. In particular, given the set of reward functions discussed above, we learn a set of corresponding K policies π_z using the recently proposed DroQ approach [43]. As an overview, DroQ is an approach which combines data augmentation in the form of random crops [44] and implicit ensembling via dropout to allow for robust, sample-efficient image based learning. We refer the reader to Hiraoka et al. [43] for a detailed description of DroQ and our project website for additional details on architecture choice and training hyperparameters.

B. Multi-Task Learning without Oracles

After attempting a particular task, the agent must decide which task to attempt next, which depends on the current situation (e.g., if it drops the hose connector, it should try to grasp it again, but if it is still holding it, it can attempt the insertion). In order to infer which task the agent should execute, we allow the user to provide next milestone labels (labels of what discrete milestone z_k should be attempted at a particular state s_k) to train a task dynamics model by performing supervised learning over the milestones and labels provided at the beginning of training. That is, given a dataset of $\mathcal{D} = \{(z_k, s_k)_{k=0}^N\}$, we can recover $p^{\phi}_{\text{task}} = \arg\max_{\phi} \mathbb{E}_{z_k, s_k \sim \mathbf{D}} \left[\log p^{\phi}_{\text{task}}(z_k|s_k)\right]$. These labels represent which task to perform upon success (e.g., the success examples for hooking the rope could be labeled with the 'unhook' task). During autonomous training, the agent samples a task from this learned model, which is then used to execute the corresponding policy π_z in the environment. Rather than re-sample this task indicator at every step, we sample it every T steps and keep it fixed during data collection. This scheme explicitly separates the task inference, and task learning allows each "sub-problem" to be treated effectively as a separate MDP.

Fig. 5: Filmstrip of the final learned hooking (left) and insertion behavior (right). Using the user-provided milestones, our robot learns a set of skills that allows it to autonomously practice hooking and unhooking (left, right two images) and recover from failure (e.g., after dropping the hook) by regrasping and reorienting the hook (left, left two images). Similarly, using the user-provided milestones, our robot learns a set of skills (e.g., grasp, insert), which together enable successful insertion (right, rightmost image) as well as the stages needed to practice autonomously (right, left three images). After 36 hours of unattended training, our system hooks onto the handle with around a 95% success rate and successfully inserts with around 80% success rate.

C. Algorithm Summary and Implementation Details

To summarize, given the milestone graph provided by the user, our system, AVAIL (Autonomy ViA mILestones), proceeds as follows. First, AVAIL performs supervised learning of the next task transitions provided by the user as described in Sec. V-B to learn $p^{\phi}_{\text{task}}(z|s)$. Next, during training, our approach chooses the most probable task z using an observed state, which is then used to collect experience using the corresponding policy π_z. We train a set of separate policies π_z for each of the K sets of example images, with separate critics Q_z and replay buffers $\mathcal{B}_z$. We parameterize each policy π_z as a deep neural network, and train each policy using the soft actor-critic algorithm (SAC) [45] using rewards R_z that are inferred via the multi-task VICE [36] algorithm trained in the loop. Finally, rather than resampling the task every step, we do so every $T = 100$ steps.

VI. Experimental Evaluation

Our experiments first aim to evaluate whether AVAIL can learn complex manipulation skills in the real world with visually indicated milestones. To do so, we evaluate our approach on three real world manipulation tasks that require successfully sequencing a set of skills and performing complex coordinated finger motions to manipulate objects. We describe first our real world evaluation followed by our evaluation in simulation which we use to provide a rigorous comparison with prior methods. The results are best viewed in the supplementary videos provided on the project website: https://sites.google.com/view/dexterous-avail/

A. Real-World Task Descriptions

We begin by describing our tasks. The objects and workspace in our experiments can be seen in Fig. 3. The environments are mostly uninstrumented, except for a passive tether that prevents the object from falling out of reach. Further task details and demonstrations can be found on the project website.

a) Using a kitchen brush.: The goal of the first task is to scrape a plate with a two sided cleaning brush (see Fig. 1). This task requires the robot to grasp the brush, and reorient it with the fingers so the bristles face the plate. The palm-down manipulation of the brush is challenging, as it requires balancing it so that it doesn't fall and rotating it around its long axis via a coordinated finger gait. The task milestones consist of grasping the brush, scraping the surface, reorienting it, and bringing the bristles in contact with the plate.

b) Hose connector insertion.: The goal of the second task is to attach a cylindrical hose connector to a peg connector, which requires reaching, grasping, reorienting, and performing the insertion as the task milestones. While the task is simpler in terms of dexterity, it requires visual perception to carefully insert the connector onto the peg connector (see Fig. 5, left).

c) Rope hooking.: The third task requires attaching a hook to a handle (see Fig. 5, right). The robot must grasp, reorient, and hook and unhook the object for its visual milestones. This task requires visually servoing the hook over a handle.

B. Real-world evaluation

To evaluate our system, we save the policies at regular intervals and evaluate their performance after training, so as not to interrupt the training process. For all tasks, the evaluation metric for each milestone is a binary success measurement based on the distance of the hand and object to the desired pose. We provide more details on our evaluation setup on our project website.

Real-world skill learning: We plot the performance of the evaluation runs as a function of the training step at which the policy was recorded in Fig. 6, providing learning curves for real-world training. Observe that AVAIL automatically provides a degree of scaffolding by successfully learning skills early on in training (blue curve in left plot, blue and orange curve in center right plot) that correspond to being able to regrasp and reorient the object. This allows the robot to continuously retry later milestones. By the end of training, we find that the robot is able to successfully perform all milestones with a $> 80\%$ success rate. We note that for all three tasks, no additional instrumentation is required beyond changing the object and fixture. Upon specifying the visual milestones, the robot is capable of completely unattended learning for approximately 48 hours of robot time.

Real-world task graph learning: Next, to understand the effect of our learned task graph, we compare to a hand-designed task graph on our insertion task. This hand-designed task graph encodes a heuristic strategy where each of the tasks are practiced sequentially. The comparative success rate can be seen in Fig. 8. We find that our learned task graph outperforms this heuristic based task graph in terms of sample efficiency. We provide additional analysis on our real world rope hooking task on our project website, in addition to simulated analysis. Overall, we find that our approach is robust to errors in our learned task graph, although future work could likely improve overall sample

Fig. 6: Success rates of different milestones across our real world dexterous manipulation tasks. Our method is able to perform all the kitchen milestones with around 80% success rate (left), successfully perform pipe insertion (middle, red curve) with around 80% success, and nearly perfectly hook and unhook the rope (right, red/purple curve). Overall, success on other milestones improves earlier in training (blue, orange curves), equipping the robot with skills to autonomously retry the task.

efficiency by improving task graph training.

C. Simulated Comparison

Finally, we compare AVAIL to prior autonomous RL method on the (`DHandValvePickup-v0`) simulation domain developed in prior work [38]. We first compare to a standard state-of-the-art RL algorithm, soft actor-critic [45] (which we denote as *SAC*) using a reward learned from example images of a successful pickup or sparse reward. Note that prior work has used this approach for real-world robotics tasks [46]. Next, we compare to a forward backward controller [47], which can be seen as providing two milestones: one to pick up the object, and one to place it back on the table. Finally, we compare to R3L [37], where we provide the "forward" policy with a set of pickup goals and follow Zhu et al. [37] by interleaving training of the forward policy with a "perturbation controller" trained with an intrinsic reward based on random network distillation [39].

Simulated comparative analysis. We evaluate the performance of each method by sampling a random initial position of the object in the workspace and running the learned policy. We evaluate the task success over the course of training in Fig. 7. Prior methods do not make successful progress on this task, due to the combination of reset-free training and the lack of a shaped reward. Without any handling of the reset-free setting, both variants of SAC fail to progress. While R3L in principle can handle the autonomous setting by perturbing the state between trials, the large, high-dimensional task simply provides too many ways for the purely novelty-seeking controller to modify the environment with a meaningful reset. The forward backward controller (red), which can be seen as an instantiation of our approach with two milestones, is the only prior method that succeeds in making progress.[3]. Overall, this suggests that the improved performance can be achieved through providing more granular milestones.

VII. Discussion and Future Work

We proposed a method for multi-task learning for dexterous manipulation from high dimensional image observations. Our method, AVAIL, constructs a task graph from a modest number of user-provided milestone examples. This task graph illustrates how to practice and reset the task,

[3]We additionally compared the forward backward controller to our approach in the real world, but found it did not make progress on our tasks. We provide details of this analysis on our project website.

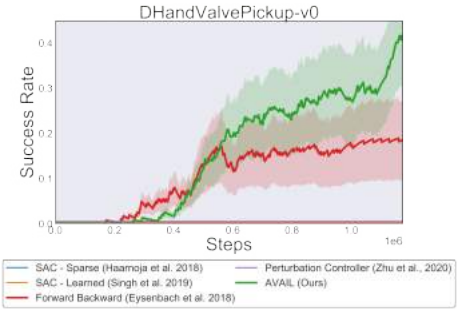

Fig. 7: Success rates of each method averaged over 5 seeds for the full task on the `DHandValvePickup-v0` domain. Novelty based resets (purple) fail to make progress in this high DoF control problem. Compared to methods with fewer degree of supervision, $K = 0, 1, 2$ milestones (blue, orange, red), our results illustrate the benefits of milestone supervision.

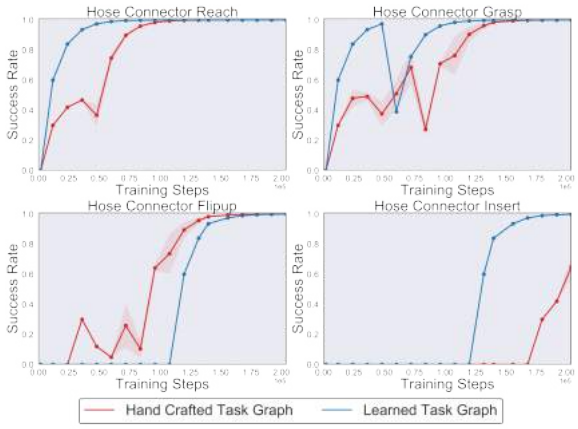

Fig. 8: A comparison of the success rate of each task on our real world insertion task. We find that using a learned task graph results in faster convergence on our real world robotic task compared to a handcoded heuristic based task graph. We find the robot begins to consistently perform the final insertion task 25% faster than a hand-coded task graph.

and provides guidance to the learning process in lieu of more standard manual reward shaping. While the milestone examples require human effort to provide, we expect in many cases that this effort is significantly lower than providing full demonstrations. Much like a teacher or coach might instruct a student not just by telling them the *goal* of a task but how they should go about practicing it, the milestone examples serve to provide guidance to the agent for how it should go about learning the desired behavior. Our experiments show that this approach effectively produces a learning process where the agent first practices the easier tasks, and then builds up the more complex tasks on top of them, all the while learning autonomously without resetting.

One limitation of our current system is that we do employ

some physical instrumentation, by tethering the object so that it doesn't fall out of reach. We found that learning to pick up the object from any location was still too challenging for our method, because the range of possible situations was too large, and developing more capable RL methods that can address this is an important direction for future work.

VIII. Acknowledgement

This research project was partially supported by the Office of Naval Research, with computing resources donated by Microsoft Azure.

References

[1] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[2] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[3] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. W. Pachocki, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *CoRR*, vol. abs/1808.00177, 2018. [Online]. Available: http://arxiv.org/abs/1808.00177

[4] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[5] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[6] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *arXiv preprint arXiv:1811.06711*, 2018.

[7] V. Kumar, A. Gupta, E. Todorov, and S. Levine, "Learning dexterous manipulation policies from experience and imitation," *arXiv preprint arXiv:1611.05095*, 2016.

[8] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine, "Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention," *arXiv preprint arXiv:2104.11203*, 2021.

[9] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in *Conference on Robot Learning*, 2020, pp. 1101–1112.

[10] I. Mordatch, Z. Popović, and E. Todorov, "Contact-invariant optimization for hand manipulation," in *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, 2012, pp. 137–144.

[11] V. Kumar, Y. Tassa, T. Erez, and E. Todorov, "Real-time behaviour synthesis for dynamic hand-manipulation," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6808–6815.

[12] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Advances in neural information processing systems*, vol. 21, 2008.

[13] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[14] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[15] K. Lowrey, S. Kolev, J. Dao, A. Rajeswaran, and E. Todorov, "Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system," in *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. IEEE, 2018, pp. 35–42.

[16] A. Allshire, M. Mittal, V. Lodaya, V. Makoviychuk, D. Makoviichuk, F. Widmaier, M. Wüthrich, S. Bauer, A. Handa, and A. Garg, "Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger," *arXiv preprint arXiv:2108.09779*, 2021.

[17] H. Van Hoof, T. Hermans, G. Neumann, and J. Peters, "Learning robot in-hand manipulation with tactile features," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 121–127.

[18] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, "Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3651–3657.

[19] D. Jain, A. Li, S. Singhal, A. Rajeswaran, V. Kumar, and E. Todorov, "Learning deep visuomotor policies for dexterous hand manipulation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3636–3643.

[20] P. Mandikal and K. Grauman, "Learning dexterous grasping with object-centric visual affordances," *arXiv preprint arXiv:2009.01439*, 2020.

[21] I. Akinola, J. Varley, and D. Kalashnikov, "Learning precise 3d manipulation from multiple uncalibrated cameras," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4616–4622.

[22] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive UAV control in cluttered natural environments," in *2013 IEEE International Conference on Robotics and Automation*, 2013.

[23] S. Reddy, A. D. Dragan, and S. Levine, "Sqil: Imitation learning via reinforcement learning with sparse rewards," *arXiv preprint arXiv:1905.11108*, 2019.

[24] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *AAAI*. AAAI Press, 2008.

[25] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum entropy deep inverse reinforcement learning," *arXiv preprint arXiv:1507.04888*, 2015.

[26] N. D. Ratliff, J. A. Bagnell, and M. Zinkevich, "Maximum margin planning," in *Machine Learning, Proceedings of the Twenty-Third International Conference ICML*, 2006.

[27] D. P. Losey and M. K. O'Malley, "Including uncertainty when learning from human corrections," in *Conference on Robot Learning*. PMLR, 2018, pp. 123–132.

[28] Y. Cui and S. Niekum, "Active reward learning from critiques," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6907–6914.

[29] J. D. Co-Reyes, A. Gupta, S. Sanjeev, N. Altieri, J. DeNero, P. Abbeel, and S. Levine, "Guiding policies with language via meta-learning," *CoRR*, vol. abs/1811.07882, 2018. [Online]. Available: http://arxiv.org/abs/1811.07882

[30] V. Myers, E. Biyik, N. Anari, and D. Sadigh, "Learning multimodal rewards from rankings," in *Conference on Robot Learning*. PMLR, 2022, pp. 342–352.

[31] D. S. Brown, W. Goo, and S. Niekum, "Better-than-demonstrator imitation learning via automatically-ranked demonstrations," in *Conference on robot learning*. PMLR, 2020, pp. 330–359.

[32] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/e6384711491713d29bc63fc5eeb5ba4f-Paper.pdf

[33] A. Levy, R. P. Jr., and K. Saenko, "Hierarchical actor-critic," *CoRR*, vol. abs/1712.00948, 2017. [Online]. Available: http://arxiv.org/abs/1712.00948

[34] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz, "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, 2012, pp. 391–398.

[35] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018.

[36] J. Fu, A. Singh, D. Ghosh, L. Yang, and S. Levine, "Variational inverse control with events: A general framework for data-driven reward definition," *arXiv preprint arXiv:1805.11686*, 2018.

[37] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine, "The ingredients of real-world robotic reinforcement learning," *arXiv preprint arXiv:2004.12570*, 2020.

[38] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine, "Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention," *arXiv preprint arXiv:2104.11203*, 2021.

[39] Y. Burda, H. Edwards, A. J. Storkey, and O. Klimov, "Exploration by random network distillation," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. [Online]. Available: https://openreview.net/forum?id=H1lJJnR5Ym

[40] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[41] K. Srinivasan, B. Eysenbach, S. Ha, J. Tan, and C. Finn, "Learning to be safe: Deep rl with a safety critic," *arXiv preprint arXiv:2010.14603*, 2020.

[42] K. Li, A. Gupta, A. Reddy, V. H. Pong, A. Zhou, J. Yu, and S. Levine, "Mural: Meta-learning uncertainty-aware rewards for outcome-driven reinforcement learning," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 6346–6356. [Online]. Available: https://proceedings.mlr.press/v139/li21g.html

[43] T. Hiraoka, T. Imagawa, T. Hashimoto, T. Onishi, and Y. Tsuruoka, "Dropout q-functions for doubly efficient reinforcement learning," *arXiv preprint arXiv:2110.02034*, 2021.

[44] I. Kostrikov, D. Yarats, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," *arXiv preprint arXiv:2004.13649*, 2020.

[45] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[46] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, "End-to-end robotic reinforcement learning without reward engineering," *arXiv preprint arXiv:1904.07854*, 2019.

[47] B. Eysenbach, S. Gu, J. Ibarz, and S. Levine, "Leave no trace: Learning to reset for safe and autonomous reinforcement learning," *arXiv preprint arXiv:1711.06782*, 2017.

[48] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," *arXiv preprint arXiv:2004.14990*, 2020.

APPENDIX

IX. REAL WORLD ENVIRONMENT DESCRIPTIONS

In the following sections, we describe the details of our real world tasks. We provide details related to experimental setup and describe our success criteria. Finally, we describe the supervision we provide the agent.

A. Object, Arena Dimensions, and Safety Considerations

The objects used in our manipulation task are a 3-D printed pipe and hook that were custom designed. Their dimensions are shown in the technical drawing in Fig. 9. All the objects are manipulated in an arena of overall size 33" × 33" consisting of a base of 20" × 20" and 8" × 8" panels. Importantly, the fixtures we use, which can be seen in the example milestone images below, are made of a flexible foam. This is in order to ensure that the robot does not place excessive forces on either the object, hand, or fixture. We leave addressing these safety considerations to future work.

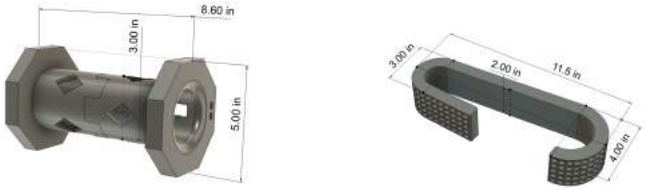

Fig. 9: Technical drawing of objects (Left: hose connector, Right: hook) used for real-world experiments, with dimensions in inches.

B. Hose Connector Insertion Task & Evaluation Criteria

The goal of this task is to have the hand reach, grasp, reorient, and insert a hose connector into an insertion point. The hose connector is attached to a fixed point at the top of the 20in × 20in arena by a rope that is 31cm long.

For this environment, we collected 300 milestone images ($84 \times 84 \times 6$) for each subtask. When training the VICE classifiers, we apply data augmentation using a random crop on the provided milestone images in addition to a randomly sampled $\mathcal{N}(0, 0.02)$ noise vector on the state.

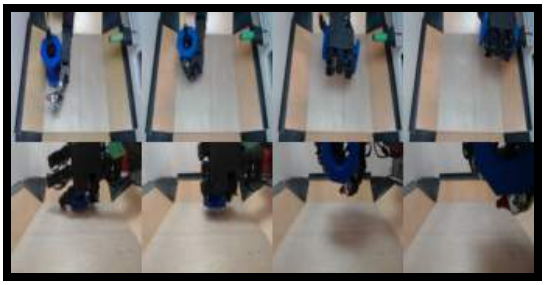

Fig. 10: Sample milestone images for Reach, Grasp, Flipup, Insert (each column respectively))

In the following table, we summarize the success criteria that we use in our experiments.

Task	Success Criteria
Reach	$\mathbb{1}\left\{\lvert x_{palm} - x_{hose}\rvert < 0.05 \text{ AND } \lvert y_{palm} - y_{hose}\rvert < 0.01\right\}$
Grasp	$\mathbb{1}\left\{ is\_held\_during\_flipup \right\}$
FlipUp	$\mathbb{1}\left\{\lvert\theta_{hose} - \theta_{goal}\rvert \leq 5^\circ\right\}$
Insert	$\mathbb{1}\left\{\lvert x_{hose} - x_{peg}\rvert < 0.05 \text{ AND } \lvert y_{hose} - y_{peg}\rvert < 0.03\right\}$

In this environment, the evaluation criteria for successful finger grasps is intuitively defined as whether or not the grasp is firm enough for performing subsequent tasks, i.e. the hose connector does not fall from the hand. θ_{hose} is the Euler angle measurement of the hose connector, where the optimal insertion angle is $\theta_{goal} = 90^\circ$. x_{hose}, x_{peg}, y_{hose}, y_{peg} are the center of mass xy-coordinate of the hose connector and the insertion peg, measured in meters.

C. Rope Hooking Task & Evaluation Criteria

The goal of this task is to have the hand grasp, reorient, hook, and unhook a carabiner hook onto a latch. The hook is attached to a fixed point at the top of the arena by a rope that is 31cm long.

For this environment, we also collect 300 milestone images ($84 \times 84 \times 6$) for each subtask. Similar to the insertion task, when training the VICE classifiers, a random crop is used as data augmentation for the milestone images and a random $\mathcal{N}(0, 0.02)$ noise is added to the milestone states.

Task	Success Criteria
Grasp	$\mathbb{1}\left\{ is\_held\_during\_flipup \right\}$
FlipUp	$\mathbb{1}\left\{\lvert\theta_{hook} - \theta_{goal}\rvert \leq 5^\circ\right\}$
Hook	$\mathbb{1}\left\{(x_{hook} - x_{latch}) > 0.01\right\}$
Remove	$\mathbb{1}\left\{(x_{hook} - x_{latch}) < 0.05\right\}$

Similar to the hose insertion task, in this environment, the evaluation criteria for successful finger grasps is intuitively defined as whether the grasp is firm enough for performing subsequent tasks, i.e. the hook object does not fall from the hand. θ_{hook} is the Euler angle measurement of the hook object along the side of its flat handle, where the ready-to-hook angle is $\theta_{goal} = 90^\circ$. x_{hook} and x_{latch} is the center of mass x-coordinate of the hook object and the latching bar, measured in meters.

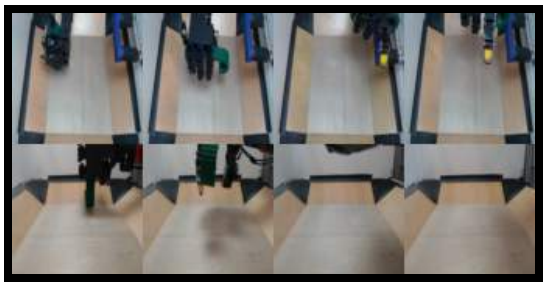

Fig. 11: Sample milestone images for Grasp, Flipup, Hook, Unhook task, (each column respectively)

D. Kitchen Cleaning Task & Evaluation Criteria

This cleaning environment demonstrates our method and the robot's ability to perform challenging tasks in a real-world kitchen setup. The robot needs to grasp the dish-washing brush, scrape the plate with the plastic end, rotate the brush 180 degrees in the hand with the palm facing downward, and clean the plate. The brush is attached to a fixed point at the top of the arena by a rope that is 31cm long. We use the same reward learning configurations as in the other two experiments.

Task	Success Criteria
Grasp	$\mathbb{1}\left\{ is\_held\_during\_flipup \right\}$
Scrape	$\mathbb{1}\left\{z_{scraper} - z_{plate} = 0\right\}$
Rotate	$\mathbb{1}\left\{165^\circ \leq \lvert\theta_{start} - \theta_{end}\rvert \leq 195^\circ\right\}$
Brush	$\mathbb{1}\left\{(z_{bristle} - z_{plate}) = 0\right\}$

In this environment, the success criteria for grasping the dish-washing brush is determined by whether the brush handle stays in hand when performing subsequent tasks. θ_{start} and θ_{end} are the Euler angle measurements of the brush object along its longitudinal (roll) axis before and after the in-hand rotation task. The goal is to rotate the side with the bristle from pointing upward to facing the plate. $z_{scraper}$ and $z_{bristle}$ are the z-coordinates of the plastic scraper and the bristle on the brush. The goal for both the scrape and brush tasks is to apply these functional sites onto the surface of the plate.

Fig. 12: Sample images for successful Grasp, Scrape, Rotate, Brush task, (each column respectively)

X. Simulated Environment Descriptions

We first describe the details of simulated environments used in the paper, and next list the hyperparameters used to train all the agents. The environment we use is based on the MuJoCo 2.0 simulator.

A. Valve3 Task

The `DHandValve3` environment contains a green, three-pronged valve placed on top of a square arena with dimensions 0.55m x 0.55m. The valve contains a circular center, and each of its prongs has an equal length of 0.1m. There are three phases in this task: reach, reposition, and pickup.The reach phase's success criteria is when the hand is within 0.1m from the valve. In the reposition phase, the success criteria for the hand is to reach for the valve, grasp the valve with its fingers, and drag the valve to within 0.1m of the arena center. Finally, success of the pickup phase is measured if the object is picked up and brought within 0.1 m of the target location which is 0.2m above the table. In order to prevent the object from falling off the table, the object is constrained to a 0.15m radius by a string from the center of the table.

The observation space of the environment is two camera views of the robot, resized to $84 \times 84 \times 3$. In addition, the proprioceptive state of the arm is provided, which is comprised of a 16-dim hand joint position, 7-dim Sawyer arm state, and 6-dim vector representing the end-effector position and euler angle. The time horizon we use in each phase of the environment is $T = 100$. We provide 300 goal images per phase, which is comparable to the number used in prior work [36].

Below, we provide the oracle task graph for the valve environment, which we use to evaluate the relative performance of our learned task graph model.

Algorithm 1 Valve3 Task Graph (Oracle Baseline)

Require: Object position $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$, previous task ϕ

Let $\begin{bmatrix} x_{center} \\ y_{center} \end{bmatrix}$ be the center coordinates of the arena (relative to the Sawyer base).
Let $\begin{bmatrix} x_{hand} \\ y_{hand} \end{bmatrix}$ be the location coordinates of the hand (relative to the arena).
$is\_centered = || \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_{center} \\ y_{center} \end{bmatrix} || < 0.1$
$is\_hand\_over\_object = || \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x\_hand \\ y\_hand \end{bmatrix} || < 0.15$
if NOT $is\_centered$ and $is\_hand\_over\_object$ **then**
    Reposition
else if NOT $is\_centered$ and NOT $is\_hand\_over\_object$ **then**
    Reach
else if $is\_centered$ **then**
    Pickup
end if

XI. Algorithm Details

In this section, we describe details related to our RL learning algorithms and also provide hyperparameters for each method.

A. AVAIL

Algorithm 2 AVAIL (Autonomy ViA mILestones)

Given: K tasks with examples states $\mathbf{D} := \{D_z, y_z\}_{z=0}^{K-1}$, start state s_0.
Train task graph $p_{taskdyn}(z|s)$ using $\mathbf{D}$
Initialize π_z, $p_z^o(o|s)$, Q_z, $\mathcal{B}_z$ for $z \in \{0, 1, \ldots, K-1\}$
for iteration $n = 1, 2, ...$ **do**
    Select current task z by querying learned task graph at the current state: $z = \text{argmax}_z p(z|s)$
    for iteration $j = 1, 2, ..., T$ **do**
        Execute π_z in environment, storing data in the buffer $\mathcal{B}_z$
        Update the current task's policy and value functions π_z, Q_z using samples from $\mathcal{B}_z$, assigning reward based on $p_z^o(o|s)$ using SAC [45].
        Update the classifier parameters, using D_z and samples from $\mathcal{B}_z$, using the VICE [36].
    end for
end for

B. Reinforcement Learning from Images

For completeness, here we describe our procedure of performing image based RL, which, as noted in prior work, presents significant optimization challenges [44], [48]. In order to make learning more practical, we make use of a combination of data augmentation techniques during training, which has been previously shown to improve image based reinforcement learning [44], and dropout regularization [43]. For all approaches we evaluate on in this work we make use of random shifts perturbations, which pad the image observation with boundary pixels before taking a random crop.

We denote $s_{aug} \sim f(s)$ as an randomly augmented image from a distribution f. We compute Q-Learning by computing the Q value for a state (s_i) over M independent augmentation. For each q function, $Q_\theta(s, a)$, we follow [43] and apply dropout followed by layer normalization in the fully connected layers of the critic.

$$\mathbb{E}_{\substack{s_i \sim B \\ a \sim \pi(\cdot|s)}} [Q_\theta(s, a)] \approx \frac{1}{M} \sum_{m=1}^{M} Q_\theta(f(s_i), a_m) \quad \text{where } a_m \sim \pi(\cdot|f(s_i)),$$

and computing a target value over L augmentations

$$y_i = r_i + \gamma \frac{1}{L} \sum_{l=1}^{L} Q_\theta(f(s_i', \nu_{i,l}'), a_{i,l}') \tag{3}$$

$$\text{where } a_{i,l}' \sim \pi(\cdot|f(s_i', \nu_{i,l}')). \tag{4}$$

This leads to a final learning rule

$$\theta \leftarrow \theta - \lambda_\theta \nabla_\theta \frac{1}{N} \sum_{i=1}^{N} (Q_\theta(f(s_i, \nu_i), a_i) - y_i)^2. \tag{5}$$

C. Hyperparameters

Here we describe the individual prior methods we compare to in detail for the purpose of reproducibility. For shared parameters, we summarize them below and provide baseline specific parameters separately.

For the simulated experiments, we train a classifier with an identical architecture to our success classifiers. During training we sample a new task a horizon of $T = 100$, as we found that in simulation the simpler naïve task graph performs comparably to the learned task graph, in real world training we employ the naïve task graph as it is arguably simpler. We provide additional experiments in the following section using the learned task which demonstrates that the learned task graph actually outperforms the naïve task graph on the more challenging real world domains.

Shared RL Hyperparameters	Value
Base Encoder	Conv(3, 3, 32, 2) 3 × Conv(3, 3, 32, 1)
Actor Architecture	FC(256, 256) FC(256, 22)
Critic Architecture	FC(256, 256) FC(256, 1)
Optimizer	Adam
Learning rate	{3e-4}
Discount γ	0.99
Target Update Frequency	1
Actor Update Frequency	1
Batch size	256
Classifier batch size	256

Shared Classifier Hyperparameters	Value
Optimizer	Adam
Learning rate	{3e-4}
Classifier steps per iteration	1
Mixup Augmentation α	10
Label Smoothing α	0.1
Classifier Architecture	Conv(3, 3, 32, 2) 3 × Conv(3, 3, 32, 1) 3 × FC(512) → ReLU() → Dropout(0.5) FC(1)

XII. Additional Real World Comparisons

We additionally provide real world comparisons that mirror the more extensive comparisons done in simulation. When compared to prior methods, we find that our simulated results are corroborated by our real world experiments. We find that our approach outperforms the forward backward algorithm on our hose insertion and hook tasks. In evaluating the efficacy of our learned task, we find, however, that on the more challenging real world tasks, our method substantially performs a naíve task graph. We provide additional details here.

A. Real World Comparisons to the Forward-Backward Controller [47]

In real world setting, we experiment and evaluate both our method and the Forward Backward (Eysenbach et al. 2018) method. For the Forward Backward method, we remove the training of the reach task by scripting it, making the learning problem easier. Additionally, we script the reach subtask, combine grasp and flipup into one forward task with horizon $T = 100$ (50 for each task) and 600 milestone images, and train insertion as the backward task. Even when making the learning problem easier for the Forward Backward method setting by scripting the reach subtask, our method outperforms the Forward Backward method. The discrepancies in the performance of the algorithms demonstrate the improvements in learning capacity as the granularity of the division of tasks increases.

For the Forward Backward method, we combine grasp, flipup, and hook into one forward task with horizon $T = 200$ (50 for grasp and flipup, 100 for hook) and 900 milestone images, and train unhook as the backward task. Once again, since our method outperforms the Forward Backward method, the results highlight the improvements in learning capacity as the granularity of the division of tasks increases.

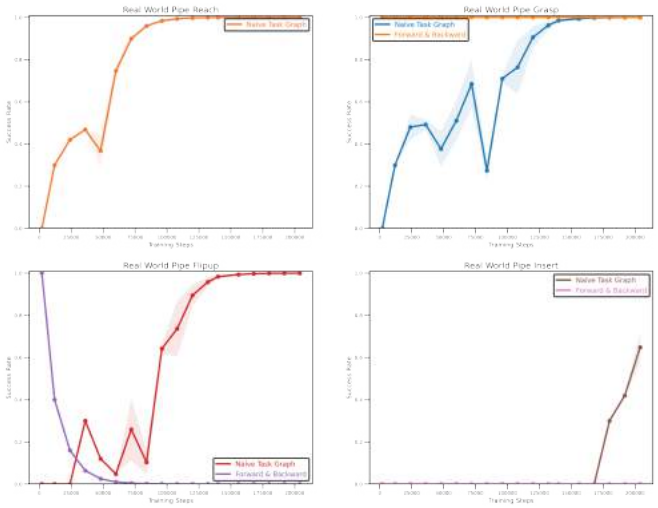

Fig. 13: Success rate of each subtask on our real world insertion task. The reach task curve with Forward Backward method is omitted as mentioned above. The Forward Backward method is unable to learn the flipup and insertion task while our method with the naïve task graph achieves substantial learning progress across all tasks.

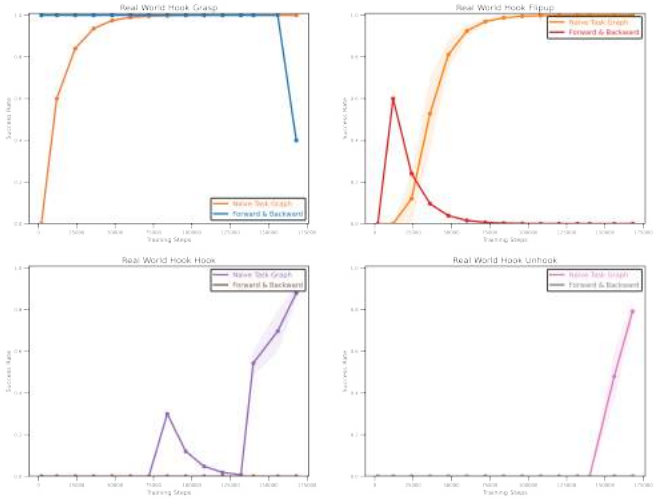

Fig. 14: Success rate of each task on our real world hooking task. The Forward Backward method is unable to learn the flipup, hook, and unhooking subtasks while our method with the naïve task graph achieves substantial learning progress across all tasks.

B. Comparison with Learned Task Graph

Here we show the success rate of our approach using a learned task graph on our real world insertion task. Different from our simulated domain, we find that on our more challenging real world tasks, we obtain significantly faster convergence (approximately 150,000 steps vs. 200,000 steps) in terms of final insertion performance using our learned task graph.

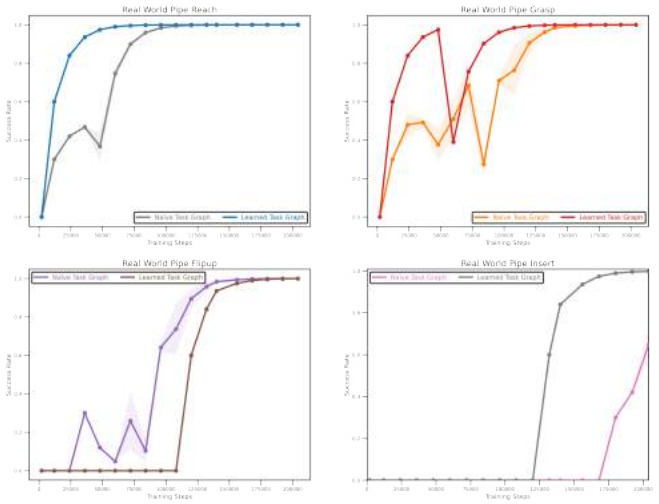

Fig. 15: Success rate of each task on our real world insertion task. We find that using a learned task graph results in faster convergence on our real world robotic task, where the robot begins to consistently perform the task around 25% faster than the naïve task graph.

Here we show the success rate of our approach using a learned task graph on our real world hook task. Different from our simulated domain, we find that on our more challenging real world tasks, we obtain significantly faster convergence (approximately 130,000 steps vs. 160,000 steps) in terms of final insertion performance using our learned task graph.

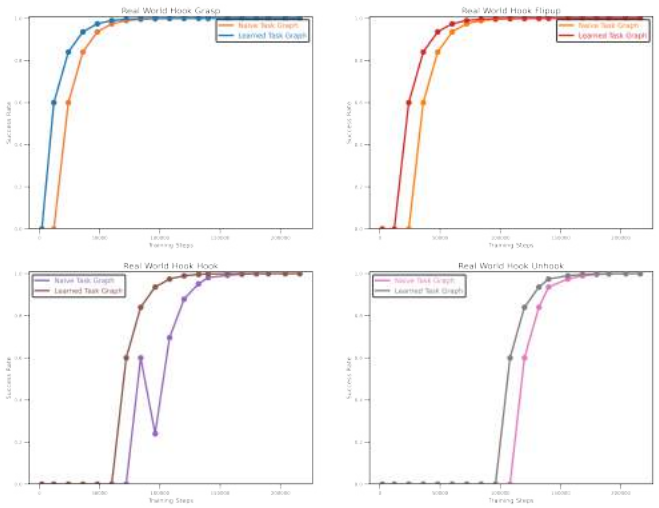

Fig. 16: Success rate of each task on our real world insertion task. We find that using a learned task graph results in faster convergence on our real world robotic task, where the robot begins to consistently perform the task around 18% faster than the naïve task graph.

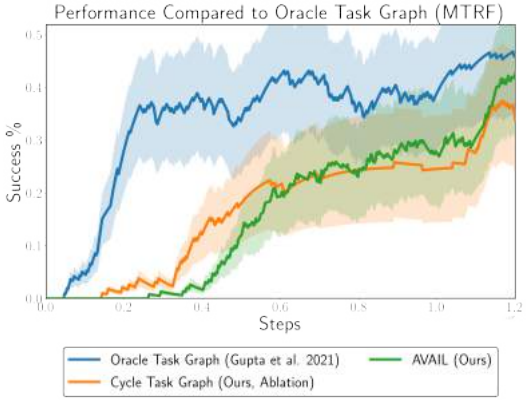

Fig. 18: A comparison of success rate on the simulated `DHandValvePickup-v0` domain compared to an oracle task graph. We find that our framework is robust to "errors" in the task graph compared to a hand crafted oracle. Both a learned task graph and ablation perform similarly given enough training.

C. Examples of Simulated Milestones

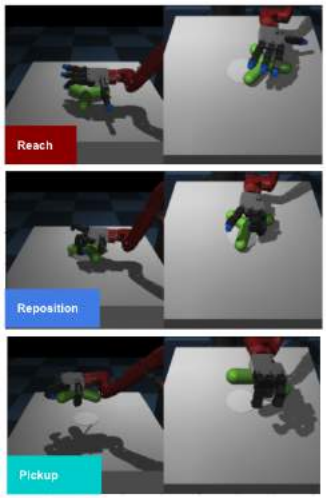

Fig. 17: The experimental domain `DHandValve-v0` we study in this work. We consider a task where the simulated robot hand is required to pick up a three-pronged object. Our observations consist of images from two viewpoints (shown above), in addition to the robot's proprioceptive state. We assume no access to a ground truth reward function, nor to episodic resets. The labels in the bottom left corners were overlayed for visualization purposes.