SPIRE: Synergistic Planning, Imitation, and Reinforcement for Long-Horizon Manipulation

Anonymous Author(s)

Abstract: Robot learning has proven to be a general and effective technique for programming manipulators. Imitation learning is able to teach robots solely from human demonstrations but is bottlenecked by the capabilities of the demonstrations. Reinforcement learning uses exploration to discover better behaviors; however, the space of possible improvements can be too large to start from scratch. And for both techniques, the learning difficulty increases proportional to the length of the manipulation task. Accounting for this, we propose SPIRE, a system that first uses Task and Motion Planning (TAMP) to decompose tasks into smaller learning subproblems and second combines imitation and reinforcement learning to maximize their strengths. We develop novel strategies to train learning agents when deployed in the context of a planning system. We evaluate SPIRE on a suite of long-horizon and contact-rich robot manipulation problems. We find that SPIRE outperforms prior approaches that integrate imitation learning, reinforcement learning, and planning by 35% to 50% in average task performance, is 6 times more data efficient in the number of human demonstrations needed to train proficient agents, and learns to complete tasks nearly twice as efficiently. View https://sites.google.com/view/spire-corl-2024 for more details.

Keywords: Reinforcement Learning, Manipulation Planning, Imitation Learning

1 Introduction

Reinforcement Learning (RL) is a powerful tool that has been widely deployed to solve robot manipulation tasks [1, 2, 3, 4]. The RL trial-and-error process allows an agent to automatically discover solutions to a task and improve its behavior over time. However, in practice, it often relies on careful reward engineering to guide the exploration process [5, 6]. The exploration burden and reward engineering problem is even more challenging to overcome for long-horizon tasks, where an agent must complete several subtasks in sequence in order to solve the task [7].

Imitation Learning (IL) from human demonstrations [8, 9] is a popular alternative to reinforcement learning. Here, humans teleoperate robot arms to collect task demonstrations. Then, policies are trained using the data. This alleviates the burden of reward engineering, since correct behaviors are directly specified through demonstrations. This paradigm has recently been scaled up by collecting large datasets with teams of human operators and robots and shown to be effective for different real-world manipulation tasks [10, 11, 12]. While these agents can be effective, they typically are imperfect, with respect to both success rates and control cost, and not robust to different deployment conditions, especially when it comes to long-horizon tasks [13].

One way to integrate the benefits of both IL and RL is to first train an agent with IL and then finetune it with RL. This can help improve the IL agent and make it robust through trial-and-error , while also alleviating the need for reward engineering due to the presence of the demonstrations. Several works have used this paradigm successfully, but long-horizon manipulation still remains challenging due to the burden of exploration and long-term credit assignment [7].

One effective approach for long-horizon manipulation is to leverage a hybrid control paradigm, where the agent is only responsible for local manipulation skills, instead of the full task [14, 15]. An example is the HITL-TAMP system [14], where an agent is trained with IL on small contact-rich segments of each tasks, and the rest of the task is performed using Task and Motion Planning

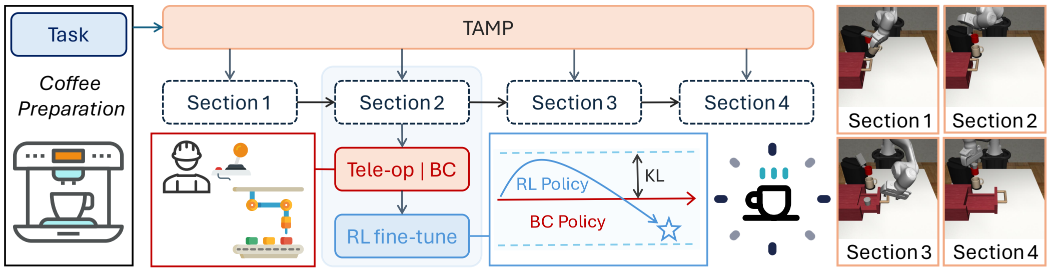


Figure 1: SPIRE Overview. (**Left**) SPIRE first attempts to solve the task with a TAMP system. When the TAMP planner encounters an action deemed too hard to plan, it then enters the *handoff* section and delegates the action to a human teleoperator to manually complete it. We record the trajectories from the human operators to build a demonstration dataset and train an IL policy with it. Finally, we train an RL policy to fine-tune the IL policy via warmstarting and deviation constraining. (**Right**) A preview of the four handoff sections in *Coffee Preparation* task.

(TAMP) [16]. Another related approach is PSL [15], which learns an agent using RL instead of IL on small segments, and uses motion planning to sequence the learned skills together. These approaches are effective for challenging long-horizon manipulation tasks, but they still often do not train perfect policies, suffering from some of the pitfalls of IL and RL. In this paper, we take inspiration from these approaches and create a hybrid control learning framework that allows for efficient imitation learning and RL-based finetuning of agents to address long-horizon manipulation tasks.

We introduce Synergistic Planning Imitation and REinforcement (SPIRE), a system for solving challenging long-horizon manipulation tasks through efficient imitation learning and RL-based fine-tuning. SPIRE decomposes each task into local manipulation segments that are learned with a policy and segments handled by a TAMP system. The manipulation segments are first trained via imitation learning and then finetuned with reinforcement learning. Our approach on 9 challenging manipulation tasks reaches an average success rate of 87.8%, vastly outperforms TAMP-gated IL [14] (52.9%) and RL [15] (37.6%). In the subset of tasks where SPIRE and IL both reach a high success rate, our method only uses 59% of the steps required by IL to complete the task. In *Tool Hang*, SPIRE fine-tunes an IL policy with only 10% success rate to 94%. We perform a thorough analysis of our method and also show that in many cases, a handful of demonstrations suffice for learning challenging tasks. Compared with IL, SPIRE improves the overall demo efficiency by 5.8 times in the evaluated subset of tasks.

Our contributions are as follows:
• We propose SPIRE, a hybrid learning-planning system that synergistically integrates the strengths of behavior cloning, reinforcement learning, and manipulation planning. SPIRE first learns a TAMP-gated policy with BC and then improves it with RL.
• We introduce key insights to enable RL-based finetuning with sparse rewards in this regime, including a mechanism to warmstart the RL process using the trained BC policy, a way to constrain exploration to be close to the BC agent behavior, and a multi-worker TAMP framework to optimize the throughput of SPIRE's RL process.
• We evaluate SPIRE on a suite of long-horizon contact-rich tasks and find that it outperforms prior hybrid learning-planning approaches in terms of **success rate averaged across tasks** (87.8%, compared to 52.9% and 37.6%), **execution efficiency** (episodes are only 59% the length of the BC agent), and **human demonstration efficiency** (6$\times$ times less data required than BC to train similar agents).

2 Related Work

Hierarchical approaches for long-horizon tasks. Hierarchical approaches decompose the challenging long-horizon tasks into easier-to-solve subtasks. RL based methods explore the division of sub-tasks with reusable skills [17, 18, 19, 20, 21]. [22, 23, 24, 25, 26] build hierarchical RL solutions with subpolicies and metacontrollers. Our work instead leverages a planner that provides guidance on which policies to learn as well as initial and terminal state distributions of tasks, compared to bottom-up HiRL methods, which tend to be data inefficient. Notably this top-down breakdown may also be achieved with a Language Model which can provide a plan composed of steps and sub-goal targets [27, 28, 29, 30, 31, 15]

Robot manipulation with demonstrations. Behavior cloning (BC) [32] learn a policy by directly mapping observations to actions, and is typically trained end-to-end using pre-collected pairs of observation and behavior data. While this is seemingly a supervised learning problem, yet in the context of robotics which add challenges. BC datasets tend to contain data sampled from *multimodal distributions*, due to intra-expert variations. Recent work address this problem implicit models including those derived from energy-based models [33, 34] or diffusion models [35, 36, 37, 38, 39]. Moreover, transformer based BC models that transformer-based categorical policies in carefully discretized action spaces do a good job handling multimodal demonstrator distributions [40, 41, 42]. Another challenge is the *correlation in sequential data*, which can lead to policies which are susceptible to temporally correlated confounders [43]. Recently several works have set out to handle this by predicting action chunks. For example, the Action Chunking Transformer (ACT) line of work [44, 45] shows that a transformer trained as a CVAE [46] to output chunks of actions performs well for a wide variety of manipulation tasks, and diffusion policy [35] shows across the board improvements when predicting action chunks. While, BC based methods combined with high-capacity models enable complex robotics tasks from demonstrations, yet challenges in robustness, and long-horizon generalization remain.

RL with experts. Experts and their demonstrations can be used to improve RL learning in multiple ways, including acting as task specifications, improving exploration, and augmenting data. Inverse RL [47, 48, 49] learns a reward model for RL from demonstrations; [50] discuss the use of demonstrations to bootstrap the learning process, followed by reinforcement learning to refine the policy; [51] warmstarts RL with a Behavior Cloning policy and grounds the Q values of non-expert actions to reduce over-optimistic estimations; [52] augments the RL replay buffer with demonstrations;[53] uses state matching for reward computation in RL. [54] shares a similar setup with ours, where they also warmstart RL with a BC policy and use a masked BC loss the constrain the RL policy from deviating. [55, 56] propose to fine-tune a semi-expert initial policy by training a residual policy on top of it with RL. However most of these works were limited in evaluation in either low-dimensional state/action or single-stage MDP settings, while we focus on building image-based agents in multi-stage sequential tasks, which poses a qualitatively different challenge.

3 Method

Our approach Synergistic Planning Imitation and REinforcement (SPIRE) learns and deploys closed-loop visuomotor skills within a TAMP system (see Fig. 1). First we frame our problem as a policy learning problem across a sequence of Markov Decision Processes (Sec. 3.1). Next, we describe our approach for incorporating both classical and learned robot skills into TAMP (Sec. 3.2) to enable TAMP-gated learning. Next, we describe how we train an initial agent with TAMP-gated Behavioral Cloning (BC) (Sec. 3.3). We then propose an RL-based finetuning algorithm to improve the BC agent with RL (Sec. 3.4). Finally, we introduce a parallelized training scheduler that is able to intelligently manage dependencies among stages when conducting RL in our setting (Sec. 3.5).

3.1 Problem Formulation

In our setup, each robot manipulation task can be decomposed into a series of alternating TAMP sections and *handoff sections*, where TAMP delegates control to a trained agent π. These sections are *TAMP-gated* [14], as they are chosen at the discretion of the TAMP system, and typically involve skills that are difficult to automate with model-based planning. We wish to train an agent π to complete these handoff sections efficiently and reliably. We model our TAMP-gated policy learning problem as a series of Markov Decision Processes (MDPs), $\mathcal{M} := (\mathcal{S}, \mathcal{A}, T, \{r^i\}, \{p_0^i\}, \gamma)_{i=1}^N$, where N is the number of MDPs (each corresponds to a TAMP handoff section), $\mathcal{S}$ and $\mathcal{A}$ are the state and action space, T is the transition dynamics, $r^i(s)$ and p_0^i are the i-th reward function and initial state distribution, and γ is the discount factor. The start and end of each handoff section is

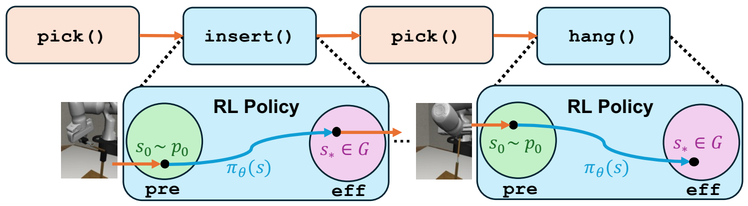


Figure 2: SPIRE execution. SPIRE computes a TAMP plan but defers execution of certain contact-rich skills, such as `insert` and `hand`, to learned agents – we call these handoff sections. The preconditions of each handoff section define the initial state distribution of the agent, and the postconditions of each action correspond to the termination states of the corresponding MDP for the handoff section.

chosen by TAMP, consequently, TAMP determines the initial state distribution p_0^i for each handoff section, and the reward function $r^i(s)$, which is a sparse 0-1 success reward based on the successful completion of the section. Our goal is to train a stochastic policy $\pi : \mathcal{S} \rightarrow \Delta_{\mathcal{A}}$ that maximizes the expected return $J(\pi) := \mathbb{E}_{(i,\tau)\sim\pi}\left[\sum_{t=0} \gamma^t r_t^i\right]$. We next describe the TAMP system.

3.2 TAMP with Learned Skills

Task and Motion Planning (TAMP) [16] is a model-based approach for synthesizing long-horizon robot behavior. TAMP integrates discrete (symbolic) planning with continuous (motion) planning to plan hybrid discrete-continuous manipulation actions. Essential to TAMP is a model of the actions that a planner can apply and how these actions modify the current state. From such a model, TAMP solvers can search over the space of plans to find a sequence of actions and their associated parameters that satisfies the task.

In SPIRE, we seek to learn a select set of TAMP actions that are impractical to manually model and then combine them with traditional actions through planning. In essence, our strategy is to learn policies π that control the system from TAMP *precondition* states to *postcondition* states.

3.3 TAMP-Gated Imitation Learning

TAMP-Gated Data Collection. We collect an initial dataset of human demonstrations through TAMP-gated human teleoperation, where the human operator collects demonstrations for handoff sections when prompted by TAMP, to form the demonstration dataset $\mathcal{D} = \{\{(s_t, a_t)_{t=1}^{H_i}, g_i\}\}$, where $s_t \in \mathcal{S}$, $a_t \in \mathcal{A}$ and H_i is the horizon, and g_i is the handoff section of the i-th trajectory. To improve the data collection efficiency, we replicate the task queuing system from [14].

Algorithm 1 SPIRE

```
1:  procedure SPIRE($G$)
2:      while True do
3:          $s \leftarrow$ OBSERVE()
4:          if $s \in G$ then
5:              return True
6:          $\vec{a} \leftarrow$ PLAN-TAMP($s, G$)
7:          for $a \in \vec{a}$ do
8:              if $a.type$ = "RL" then
9:                  $\pi \leftarrow a.policy$
10:                 EXECUTE-POLICY($\pi$)
11:                 break
12:             else
13:                 $\tau \leftarrow a.trajectory$
14:                 EXECUTE-TRAJECTORY($\tau$)
```

TAMP-Gated Behavioral Cloning. Given the dataset $\mathcal{D}$, we train a Behavioral Cloning (BC) policy parameterized by ϕ to minimize the negative log-likelihood loss over the demonstration dataset: $\phi^* = \text{argmin}_\phi \, \mathbb{E}_{(s,a)\sim\mathcal{D}}[-\log \pi_\phi(a|s)]$. The trained BC agent π_ϕ may have substantial room for improvement, depending on the complexity of the task, and the number of demonstrations available for training. We next describe our RL-based finetuning procedure (Sec. 3.4) that allows this agent to be improved through reinforcement learning.

3.4 RL Finetuning

Given a trained BC agent π_ϕ, we wish to train an RL agent π_θ to improve performance further. To avoid reward engineering, we only assume access to sparse 0-1 completion rewards for each handoff section provided by TAMP (Sec. 3.1). However, exploration in sparse-reward settings has been shown to be challenging [57, 58, 59, 60], especially in continuous state and action spaces. Fortunately, we can use the BC policy trained in the previous section as a reference point for exploration – we want to restrict the behavior of the RL policy to be in a neighborhood of the BC policy. This is achieved by a) warmstarting the RL policy optimization using the BC policy, and b) enforcing a constraint on the deviation between the RL policy and the BC policy.

Warmstarting RL optimization with BC. We tested two ways to warmstart the RL agent. *Initialization.* One method is to initialize the weights of the RL agent with those of the trained BC agent, $\theta \leftarrow \phi^*$, where $\phi^* = \arg\min_\phi L_{BC}(\phi)$, and subsequently finetune the weights with online RL objectives. Despite being easy to implement, this can be less flexible since it requires the agent structure of the RL and BC policies to match. Furthermore, researchers have found that retraining neural networks with different objectives can cause the network to lose plasticity [61], which can make the policy harder to optimize because of the objective shift from BC to RL. *Residual Policy.* An alternative way is to fix the BC policy as a reference policy and train a residual policy on top of it. Let the residual policy be $\pi_\theta^+(s)$. The residual policy shares the same action space as the normal policy but is initialized to close to zero. The final action is defined as a summation of the reference action $a \sim \pi_{\phi^*}(s)$ and the residual action $a^+ \sim \pi_\theta^+(s)$. In practice, we only add the mean of the reference policy to the residual action instead of sampling the reference action.

Constraining Deviation between BC and RL agents. The sparsity of reward signals produces high-variance optimization objectives, which can lead the RL policy to quickly drift away from BC and lose the exploration bonus from warmstarting. Therefore, it is critical to constrain the policy output to be close to the BC agent throughout the training process. We achieve this by imposing a KL-divergence penalty on the RL objective. We conclude our RL optimization objective as follows: $J_{FT}(\theta) := J(\pi_\theta) - \alpha D_{KL}(\pi_\theta \| \pi_{\phi^*})$, where $D_{KL}(p\|q) := \mathbb{E}_{(s,a)\sim p}\left[\log \frac{p(a|s)}{q(a|s)}\right]$ and α is the weight for the penalty term.

3.5 Multi-Worker Scheduling Framework

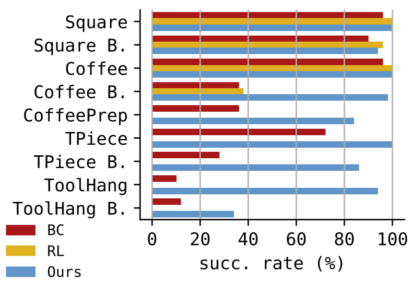
Making our TAMP-gated framework compatible with modern reinforcement learning procedures requires addressing several challenges. First, TAMP can take dozens of seconds for a single rollout, which severely lowers the throughput of RL exploration. Second, the TAMP pipeline executes each section sequentially, which means that later handoff segments can only be sampled when previous handoff segments are completed successfully. This leads to an imbalance of episodes for the different handoff segments, and is potentially problematic for the RL agent. In light of these challenges, we propose a multi-worker TAMP scheduling framework to integrate TAMP into RL fine-tuning. The framework consists of three components – a group of TAMP workers that run planning in parallel, a status pool that stores the progress of the workers, and a scheduler that distributes tasks to the workers and balances the initial states. We further describe how the framework allows for curriculum learning, and how the framework accelerates learning efficiency for RL training.

Algorithm 2 Scheduler

```
1: procedure SCHEDULER(WORKERS, STATUSQUEUE, POLICY, STRATEGY)
2:     while True do
3:         $i, j \leftarrow$ STATUSQUEUE.$pop()$
4:         if STRATEGY.$accepts(j)$ then
5:             while not WORKERS[$i$].$done()$ do
6:                 $s_{obs} \leftarrow$ WORKERS[$i$].$observe()$
7:                 $a \leftarrow$ POLICY.$act(s_{obs})$
8:                 WORKERS[$i$].$step(a)$
9:         else
10:            WORKERS[$i$].$reset()$
```

TAMP workers. Each TAMP worker has an environment instance and repeatedly runs a TAMP planner. Upon reset, the TAMP worker initiates TAMP until a handoff section has been reached. It then sends a pair (`#worker, #section`) representing its ID and which handoff section it has entered to the status queue, indicating that it is ready to take RL agent actions. The worker then enters an idle state until it receives a command from the scheduler. Depending on the command, the worker either resets itself or starts interacting with the environment by exchanging actions and states with the scheduler. If the current section has been solved, the worker sends a success signal to the scheduler and runs TAMP until it reaches the next handoff section.

Scheduler. (Algorithm 2) The scheduler is a centralized component that manages the TAMP workers. It also provides an environment abstraction to the single-threaded RL process. The scheduler is configured with a sampling strategy. Upon initialization, it first pops an item from the status queue.



Episode Duration:	**BC** [14]	**RL** [15]	**Ours**
Square	18.1	**8.3**	11.6
Square Broad	24.5	**8.4**	13.6
Coffee	63.1	**15.0**	38.4
Coffee Broad	80.6	**25.7**	61.3
Coffee Preparation	193.3	-	**168.5**
Three Piece	58.7	-	**34.0**
Three Piece Broad	62.2	-	**38.1**
Tool Hang	81.8	-	**61.7**
Tool Hang Broad	130.5	-	**109.8**

Figure 3: Full evaluation. Comparing the success rates (**left**) and the average duration (**right**) of successful rollouts of *HITL-TAMP-BC* (**BC**), *TAMP-gated Plan-Seq-Learn* (**RL**), and SPIRE (**Ours**) across all 9 tasks. Each datapoint is chosen from the best run out of 5 seeds and is averaged from 50 rollouts. SPIRE improves the BC policy in terms of both success rate and average duration in all 9 tasks and reaches 80% success rate in 8. RL has an advantage in average duration in the easier set of tasks but fails to learn anything in the rest.

According to the sampling strategy, the scheduler either rejects this section, in which case it sends a resetting signal to the corresponding worker; or starts a new episode and interacts with the worker. The status queue is a FIFO queue that stores the availabilities of the TAMP workers. It supports single-thread reading from the scheduler and multi-thread writing from the TAMP workers through a simple locking mechanism.

Curriculum Learning. The behavior of the scheduler depends on a sampling strategy, allowing it to function as a *curriculum* for the RL agent. We consider two strategies: `permissive` is the default strategy that allows all sections through, while `sequential` only accepts a section when the success rate of passing all the previous sections reaches a threshold. `sequential` allows controlling the initial state distribution during the early stages of training, to ensure the RL agent achieves proficience in each section sequentially before continuing onto the next section.

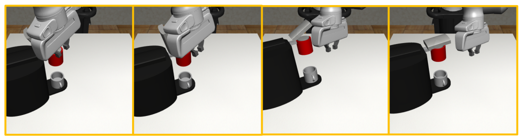
Remarks on Efficiency. Suppose a TAMP planning process takes at most T seconds over the episode; each environment interaction step, counting communication latency, takes at least t seconds; and each handoff section is at least H steps. If the number of TAMP workers $n \geq \frac{T}{tH}$, the proposed multi-worker TAMP scheduling framework reaches a throughput of at least $1/t$ frames per second. In comparison, the single-worker counterpart has a worst-case throughput of $\frac{H}{T+tH}$ frames per second. Suppose that the planning process is slower than the handoff sections by a factor k (e.g. $T = k \cdot tH$), then our framework is faster than the single-worker alternative by a factor of $k + 1$.

4 Experiments

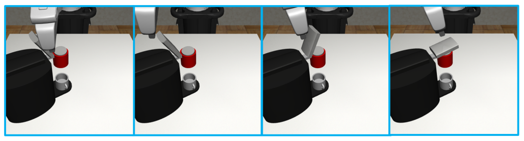
Tasks. For evaluation, we follow [14] and choose a set of long-horizon manipulation tasks, namely *Square*, *Coffee*, *Three Piece*, and *Tool Hang*. We also include the broad variants of those tasks, where we use a broad object initialization region, and *Coffee Preparation*, which has the longest horizon with four handoff sections. See the appendix for more details.

Environment Details. *Observation space.* For most tasks, we use a single 84×84 RGB image from the wrist-view camera. For *Tool Hang*, we use the front-view camera instead since the wrist-view is mostly occluded. For *Tool Hang Broad* and *Coffee Preparation*, we use both wrist-view and front-view cameras, as well as proprioception state (end-effector pose and gripper finger width). *Action space.* Actions are 7-dimensional (3-dim delta end-effector position, 3-dim delta end-effector rotation, 1-dim gripper actuation). *Horizon.* Each handoff section is limited to 100 steps (5 seconds with 20Hz control frequency) for all tasks, except for *Tool Hang Broad*, where the limit is 200 steps.

Baselines. We compare our method with two baselines: *HITL-TAMP-BC* (**BC**), which is adapted from [14] to match our network structure; and *TAMP-gated Plan-Seq-Learn* (**RL**), which is adapted from [15] by replacing the LLM-based planning system with our TAMP system for fair comparison. We collected 200 human demonstrations for each task to train the behavior cloning policy. For RL, we use DrQ-v2 [62] as the base algorithm. See the appendix for more details.



(a) Naive RL agent



(b) SPIRE agent

Figure 4: Qualitative comparison. Rollouts of vanilla RL vs our method. The first agent attempts to close the lid by knocking the coffee machine, while our agent follows the demonstrations and closes the lid with fingers.

Evaluation. We evaluate each trained agent for 50 rollouts and report the success rate and average completion steps in the successful rollouts. We train 5 seeds for each algorithm and report the best-performing agent (success rate-wise, tie-breaking with average steps) unless otherwise specified.

4.1 Results

SPIRE outperforms both TAMP-gated BC and RL. We compare our method with the TAMP-gated BC [14] and RL [15] baselines across all 9 tasks (see Fig. 3). SPIRE reaches 80% success rate in 8 out of 9 tasks, while BC and RL only reach 80% in 3 tasks each. In *Tool Hang*, our method reaches 94% success rate despite the BC counterpart only having 10%, which is over 9-times improvement. Remarkably, this low-performing BC agent is enough to help address the exploration burden (unlike RL, 0% success) and train a near-perfect agent. Across all 9 tasks, SPIRE averages a 87.8% success rate, while BC and RL only average 52.9% and 37.6% respectively.

SPIRE produces more efficient agents than BC through RL fine-tuning. SPIRE agents have lower average completion times than their BC and RL counterparts (Fig. 3, right). Even in tasks such as *Square*, *Square Broad*, *Coffee*, *Three Piece*, where BC policies already have high success rates, our method improves the efficiency by only using an average of 59% completion time.

SPIRE's use of the BC agent helps address the RL exploration burden on challenging long-horizon tasks. Exploration in RL with sparse rewards is extremely challenging, especially for robot manipulation tasks for their continuous and high-dimensional observation and action space. Our method solves the initial exploration problem by anchoring policy learning around the BC agent. As shown in Figure 3, RL policies without utilizing BC only reach nonzero success rates in *Square*, *Coffee* and their *Broad* variants, all of which have only one handoff section and relatively shorter horizons. Even in *Coffee Broad*, RL encounters exploration difficulties due to the broader object distribution, resulting in only partially solving the task.

Qualitatively, SPIRE can improve agent behavior without introducing undesirable behavior, unlike RL. Safety awareness has always been a critical matter in robotics learning. Safety constraints can be hard to define with numerical values, which adds to the challenges of realizing safety in RL. We notice that in *Coffee*, RL policy has a much shorter completion time than our method. This is at the cost of ignoring safety concerns. We compare two rollouts of RL and our method in Figure 4. The RL-trained policy attempts to close the lid by knocking the coffee machine with the arm, which can potentially damage the robot and the coffee machine and even cause danger to humans; while our method preserves safety awareness by following the demonstration's practice of closing the lid with its fingers.

SPIRE can train proficient agents using just a handful of human demonstrations. BC methods can require several human demonstrations to train proficient agents, which can be a major drawback due to the cost of collecting this data [9]. We reduce the number of human demonstrations used by SPIRE to 10 and 50 (instead of 200 as in Fig. 3), and we plot the minimum of demonstrations needed to reach at least 80% success rate in Fig. 5. As the plot shows, SPIRE can successfully fine-tune a BC policy trained with as few as 10 demos in all evaluated tasks except for *Tool Hang* and *Coffee Broad*, for which 50 demos are enough. In the 7 tasks, our method needs 150 demos in total, while BC needs more than 870, a 5.8× improvement in efficiency.

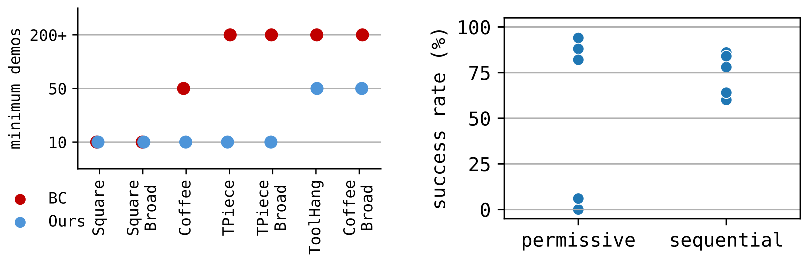


Figure 5: Demo efficiency and sampling strategy ablation. (**Left**) Minimum number of demos needed to reach at least 80% success rate. (**Right**) Success rates across 5 seeds in *Tool Hang*, comparing `permissive` and `sequential` strategies. `sequential` has a lower variance but `permissive` has the better top-1 policy.

4.2 Ablation Study

We conduct two ablative studies to investigate (1) the value of the KL-divergence penalty and (2) the value of curriculum learning, governed by the two scheduler sampling strategies `permissive` and `sequential`. In this section, we compare the performance distribution of the 5 runs instead of only the top-1 run for a more comprehensive evaluation.

Value of divergence penalty. We ablate the divergence penalty on two representative tasks, *Three Piece* & *Tool Hang* (Table 6) and observe a drastic performance drop (84% to 17.6%, 74% to 0%).

The sparsity in rewards leads to high-variance optimization objectives for RL. As a result, even when warmstarted with BC, the RL policy can quickly deviate from it, especially when the chance of reaching the reward signal is low. Therefore, constraining the policy close to BC throughout the training is critical. We select two representative tasks, *Three Piece* and *Tool Hang* for this ablation. The result is shown in Figure 6. Without the divergence penalty, the RL policies deviated immediately and never returned.

Task	w/ (%)	w/o (%)
Three Piece	**84.0** (**34.7**)	17.6 (39.4)
Tool Hang	**74.4** (**11.8**)	0.0 (0.0)

Figure 6: KL-divergence penalty ablation. Mean and standard deviation (in parenthesis) of success rates across the 5 seeds in *Three Piece* and *Tool Hang*, with and without KL-divergence penalty. In both tasks, the divergence penalty improves the performance by a large margin.

Value of curriculum learning. We compare the two sampling strategies in *Tool Hang* task. The result is shown in Figure 5. `sequential` strategy shows a much smaller variance compared with `permissive`. However, `permissive` produces the better top-1 seed performance. The main difference between the two strategies is how the second section states emerge during training. For `permissive`, the second section states emerge gradually as the success rate of passing the first section gets higher, resulting in a more gentle distributional shift that leads to a higher overall success rate; for `sequential`, the shift is more abrupt, but it gains the advantage of fewer distraction states in the early stage, resulting in a more stable training process.

5 Conclusion

We presented SPIRE, an integrated approach for deploying RL, BC, and planning harmoniously. We showed how BC can be used to not only warm-start RL but also guide the RL process via focused exploration. We introduced a scheduling mechanism to improve RL data throughput and increase learning efficiency. Finally, we evaluated SPIRE in simulation against recent hybrid learning-planning baselines and found that SPIRE results in more successful and efficient policies.

Limitations. We focus on tasks that center around object-centric manipulation of rigid objects in table-top environments. We assume that a human teleoperator can demonstrate the learned skills to warmstart RL. The TAMP component assumes that the state is observable and comprised of rigid objects, possibly connected with articulation. Our method is most effective on tasks that involve quasi-static components, as these are most effectively modeled and addressed by TAMP systems. To simplify RL training, we only considered Markovian policies; however, using neural network architectures with history, such as RNNs, may boost performance [9].

References

[1] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pages 651–673. PMLR, 2018.

[2] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[3] B. Tang, M. A. Lin, I. Akinola, A. Handa, G. S. Sukhatme, F. Ramos, D. Fox, and Y. Narang. Industreal: Transferring contact-rich assembly tasks from simulation to reality. *arXiv preprint arXiv:2305.17110*, 2023.

[4] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. Sundaralingam, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5977–5984. IEEE, 2023.

[5] D. Silver, S. Singh, D. Precup, and R. S. Sutton. Reward is enough. *Artificial Intelligence*, 299:103535, 2021.

[6] A. Y. Ng, S. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[7] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.

[8] T. Zhang, Z. McCarthy, O. Jow, D. Lee, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. *arXiv preprint arXiv:1710.04615*, 2017.

[9] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

[10] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[11] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

[12] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

[13] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[14] A. Mandlekar, C. Garrett, D. Xu, and D. Fox. Human-in-the-loop task and motion planning for imitation learning. In *7th Annual Conference on Robot Learning*, 2023.

[15] M. Dalal, T. Chiruvolu, D. Chaplot, and R. Salakhutdinov. Plan-seq-learn: Language model guided rl for solving long horizon robotics tasks. 2024.

[16] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021.

[17] P. Dayan and G. E. Hinton. Feudal reinforcement learning. In *NIPS*, 1992.

[18] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112:181–211, 1999.

[19] P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. *AAAI*, 2017.

[20] A. S. Vezhnevets, S. Osindero, T. Schaul, N. M. O. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *ICML*, 2017.

[21] O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. In *NeurIPS*, 2018.

[22] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. B. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *NIPS*, 2016.

[23] D. Lyu, F. Yang, B. Liu, and S. M. Gustafson. Sdrl: Interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *AAAI*, 2019.

[24] J. Rafati and D. C. Noelle. Learning representations in model-free hierarchical reinforcement learning. *AAAI*, 2019.

[25] S. Sohn, H. Woo, J. Choi, and H. Lee. Meta reinforcement learning with autonomous inference of subtask dependencies. *ICLR*, 2020.

[26] R. Costales, S. Iqbal, and F. Sha. Possibility before utility: Learning and using hierarchical affordances. *ICLR*, 2022.

[27] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[28] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: program generation for situated robot task planning using large language models. *Autonomous Robots*, 47(8):999–1012, 2023.

[29] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

[30] N. Yoshikawa, M. Skreta, K. Darvish, S. Arellano-Rubach, Z. Ji, L. Bjørn Kristensen, A. Z. Li, Y. Zhao, H. Xu, A. Kuramshin, et al. Large language models for chemistry robotics. *Autonomous Robots*, 47(8):1057–1086, 2023.

[31] M. Skreta, Z. Zhou, J. L. Yuan, K. Darvish, A. Aspuru-Guzik, and A. Garg. RePLan: Robotic Replanning with Perception and Language Models. *arXiv e-prints*, art. arXiv:2401.04157, Jan. 2024. doi:10.48550/arXiv.2401.04157.

[32] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[33] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[34] D. Jarrett, I. Bica, and M. van der Schaar. Strictly batch imitation learning by energy-based distribution matching. *Advances in Neural Information Processing Systems*, 33:7354–7365, 2020.

[35] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[36] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu. 3d diffusion policy. *arXiv preprint arXiv:2403.03954*, 2024.

[37] L. Chen, S. Bahl, and D. Pathak. Playfusion: Skill acquisition via diffusion from language-annotated play. In *Conference on Robot Learning*, pages 2012–2029. PMLR, 2023.

[38] Z. Xian, N. Gkanatsios, T. Gervet, T.-W. Ke, and K. Fragkiadaki. Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In *7th Annual Conference on Robot Learning*, 2023.

[39] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[40] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

[41] Z. J. Cui, Y. Wang, N. M. M. Shafiullah, and L. Pinto. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.

[42] S. Lee, Y. Wang, H. Etukuru, H. J. Kim, N. M. M. Shafiullah, and L. Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

[43] G. Swamy, S. Choudhury, J. A. Bagnell, and Z. S. Wu. Causal imitation learning under temporally correlated noise, 2022.

[44] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023.

[45] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

[46] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf`.

[47] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Neural Information Processing Systems*, 2016.

[48] J. Ho, J. K. Gupta, and S. Ermon. Model-free imitation learning with policy optimization. *ArXiv*, abs/1605.08478, 2016.

[49] C. Finn, P. F. Christiano, P. Abbeel, and S. Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *ArXiv*, abs/1611.03852, 2016.

[50] V. G. Goecks, G. M. Gremillion, V. J. Lawhern, J. Valasek, and N. R. Waytowich. Integrating behavior cloning and reinforcement learning for improved performance in dense and sparse reward environments. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 465–473, 2020.

[51] T. Hester, M. Vecerík, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. P. Agapiou, J. Z. Leibo, and A. Gruslys. Learning from demonstrations for real world reinforcement learning. *ArXiv*, abs/1704.03732, 2017.

[52] M. Vecerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. M. O. Heess, T. Rothörl, T. Lampe, and M. A. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *ArXiv*, abs/1707.08817, 2017.

[53] S. Haldar, J. Pari, A. Rai, and L. Pinto. Teach a robot to fish: Versatile imitation from one minute of demonstrations. *arXiv preprint arXiv:2303.01497*, 2023.

[54] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299, 2017.

[55] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine. Residual reinforcement learning for robot control. *2019 International Conference on Robotics and Automation (ICRA)*, pages 6023–6029, 2018.

[56] T. Silver, K. R. Allen, J. B. Tenenbaum, and L. P. Kaelbling. Residual policy learning. *ArXiv*, abs/1812.06298, 2018.

[57] B. C. Stadie, S. Levine, and P. Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *ArXiv*, abs/1507.00814, 2015.

[58] H. Tang, R. Houthooft, D. Foote, A. Stooke, X. Chen, Y. Duan, J. Schulman, F. D. Turck, and P. Abbeel. #exploration: A study of count-based exploration for deep reinforcement learning. *NIPS*, 2017.

[59] A. P. Badia, P. Sprechmann, A. Vitvitskyi, D. Guo, B. Piot, S. Kapturowski, O. Tieleman, M. Arjovsky, A. Pritzel, A. Bolt, and C. Blundell. Never give up: Learning directed exploration strategies. *ICLR*, 2020.

[60] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. First return then explore. *Nature*, 590 7847:580–586, 2021.

[61] Z. Abbas, R. Zhao, J. Modayil, A. White, and M. C. Machado. Loss of plasticity in continual deep reinforcement learning. *ArXiv*, abs/2303.07507, 2023.

[62] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

A Overview

The Appendix contains the following content.

- **Policy Learning Details** (Appendix B)): details on hyperparameters used
- **Ablation: SPIRE without TAMP** (Appendix C): ablation study on the effect of removing TAMP-gating and directly running BC and RL fine-tuning
- **Comparison to Additional Methods** (Appendix D): comparison to other RL methods that leverage demonstrations
- **Tasks** (Appendix E): details on tasks used to evaluate SPIRE
- **Variance Across Seeds** (Appendix F): discussion on the variance of results across different seeds and how results are presented

B Policy Learning Details

Table 1: DrQ-v2 hyperparameters.

Network structure	CNN
Learning rate	1e-4
Discount	0.99
Batch size	256
n-step returns	3
Action repeat	1
Seed frames	4000
Feature dim	50
Hidden dim	1024
Optimizer	Adam

Hyperparameters. The base RL algorithm for all our experiments is DrQ-v2 [62]. The specific hyperparameters are in Table 1.

Observation. For most tasks, we use one 84×84 RGB image from the wrist camera as the only observation. For *Tool Hang*, we use a front-view camera instead since the wrist-view is heavily occluded. For *Tool Hang Broad* and *Coffee Preparation*, we use both camera views plus proprioception state (end-effector pose and gripper finger width). We use the default CNN structure from DrQ-v2 to encode the image observations. For tasks with multiple observations, we first encode the image observations each with an independent CNN network, then concatenate the CNN outputs alongside the low-dimensional observations such as proprioception states to form the feature vector.

Action. All our tasks share a 7-dimensional (6-DOF delta movement of the end-effector and 1 dimension for finger control) continuous action space. The action is modeled as a normal distribution with a scheduled standard deviation.

Table 2: Comparing the success rates of *Square* and *Square Broad* with and without TAMP.

Task	**BC**	**RL**	**Ours**
Square *w/* TAMP	98%	100%	100%
Square *w/o* TAMP	2%	0%	94%
Square Broad *w/* TAMP	100%	100%	100%
Square Broad *w/o* TAMP	0%	0%	0%

C Ablation: SPIRE without TAMP

We provide an additional ablation study on the high-level planner, TAMP. To do so, we treat the whole task as one handoff section. The agent only receives a reward of one if it completes the whole task. We collect 200 full demonstrations in *Square*, train a BC policy, and apply SPIRE to fine-tune the BC policy. Since the trajectory becomes longer and the robot now needs to handle object transportation, a single local wrist-view becomes insufficient. We thus include both the wrist view and the global front view, as well as the robot proprioception states in the observation for the w/o TAMP variant. The result is shown in Table 2.

Even though the w/o TAMP variant has more information from observations, the BC and RL policies are significantly worse than the w/ TAMP counterpart. The increased horizon makes the BC policy easier to drift away to regions less frequently visited in demonstrations and makes RL exploration much harder. In *Square*, despite the low starting quality, SPIRE still fine-tunes BC to reach a 94% success rate, demonstrating the effectiveness of RL fine-tuning. However, when the initialization range increases in *Square Broad*, even SPIRE fails to find an acceptable policy.

In conclusion, TAMP (1) confines the agent-controlled section to a small local area, reducing the need for global information, and (2) decreases the horizon (11.6 w/ TAMP, 101.7 w/o TAMP in *Square*) for the learned agent, reducing compounding errors and exploration difficulty.

D Comparison to Additional Methods

In each handoff section from TAMP, SPIRE utilizes the demonstrations by training a behavior cloning agent and using RL to fine-tune it. There are alternative methods to combine expert demonstrations and RL, which can be readily plugged in as replacements to SPIRE. In this section, we make connections from our method to GAIL [47]. The discriminator-based IRL reward in GAIL serves the same purpose as our KL penalty term - preventing the current policy from deviating from the expert policy. We draw further connection by showing that our KL penalty is the same as the IRL reward function in GAIL with an alternative discriminator objective and a different reward form.

Let π_E be the expert policy. The IRL reward function in GAIL is $-\log(1 - D(s, a))$, where $D : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the discriminator that maximizes

$$J(D) := \mathbb{E}_{\tau\sim\pi}[\log(1 - D(s, a))] + \mathbb{E}_{\tau\sim\pi_E}[\log(D(s, a))] \tag{1}$$

If we use an alternative objective:

$$\hat{J}(D) := \mathbb{E}_{s\sim\pi_E, a\sim\text{Unif}}[-D(s, a)] + \mathbb{E}_{\tau\sim\pi_E}[\log(D(s, a))] \tag{2}$$

The alternative objective discriminates π_E from a fixed policy rather than the current learned policy π. Assume π_E has full support, then maximizing $\hat{J}(D)$ is equivalent to maximize for every $s \in \mathcal{S}$:

$$\begin{aligned}
\hat{J}_s(D) :=& \mathbb{E}_{a\sim\text{Unif}}[-D(s, a)] + \mathbb{E}_{a\sim\pi_E(\cdot|s)}[\log(D(s, a))] && (3)\\
=& -\left(\int D(s, a)\,\mathrm{d}a\right) + \left(\int \pi_E(a \mid s)\log(D(s, a))\,\mathrm{d}a\right) && (4)\\
=& -\left(\int D(s, a)\,\mathrm{d}a\right) + \left(\int \pi_E(a \mid s)\log\pi_E(a \mid s)\,\mathrm{d}a\right) + \left(\int \pi_E(a \mid s)\log\frac{D(s, a)}{\pi_E(a \mid s)}\,\mathrm{d}a\right) && (5)\\
=& -\left(\int D(s, a)\,\mathrm{d}a\right) + H(\pi_E(\cdot \mid s)) + \left(\int \pi_E(a \mid s)\log\frac{D(s, a)}{\pi_E(a \mid s)}\,\mathrm{d}a\right) && (6)\\
\leq& -\left(\int D(s, a)\,\mathrm{d}a\right) + H(\pi_E(\cdot \mid s)) + \left(\int \pi_E(a \mid s)\left(\frac{D(s, a)}{\pi_E(a \mid s)} - 1\right)\mathrm{d}a\right) && (7)\\
=& -\left(\int D(s, a)\,\mathrm{d}a\right) + H(\pi_E(\cdot \mid s)) + \left(\int D(s, a)\,\mathrm{d}a\right) - \left(\int \pi_E(a \mid s)\,\mathrm{d}a\right) && (8)\\
=& H(\pi_E(\cdot \mid s)) - 1 && (9)
\end{aligned}$$

where H is the entropy. (7) holds since $\log x \leq x - 1$ for all $x > 0$, and only equates when $x = 1$, i.e., $\hat{D}(s, a) = \pi_E(a \mid s)$. Since (9) is a constant, the maximum of $\hat{J}(D)$ can be taken when (7) equates, which means the optimal solution of $\hat{J}(D)$ is $\hat{D}(s, a) = \pi_E(a \mid s)$. Our KL penalty then is equivalent to using an IRL reward of $\log(\hat{D}(s, a)) = \log\pi_E(a \mid s)$.

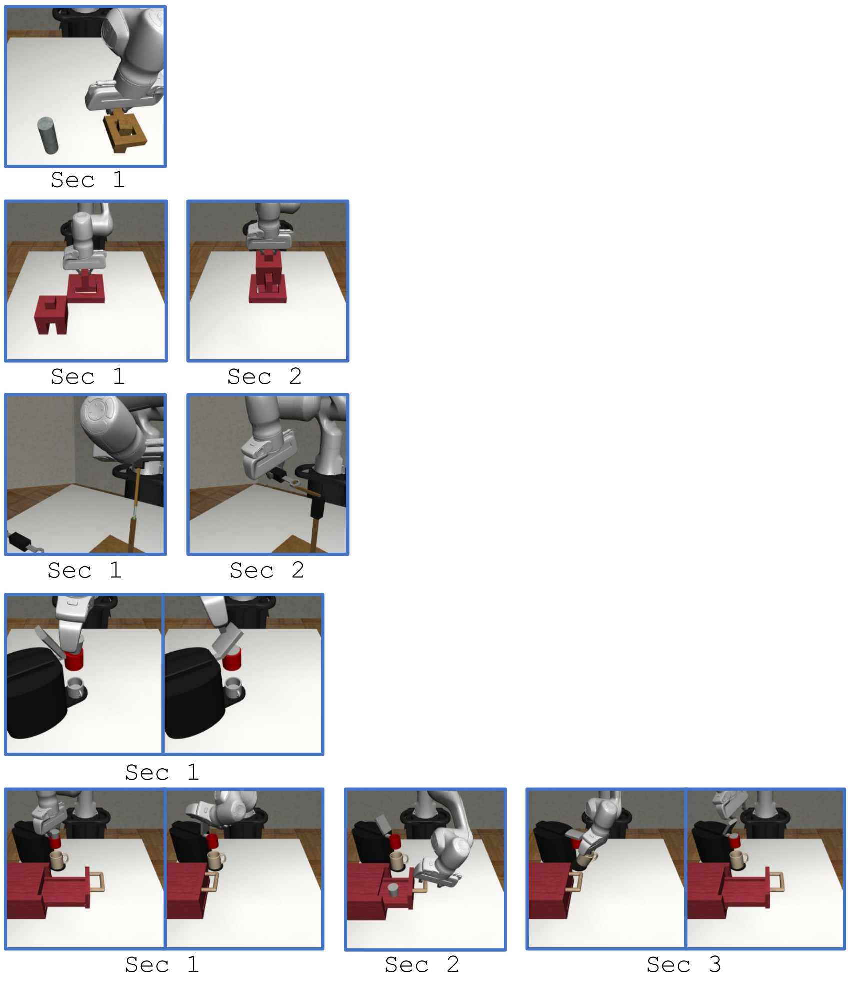


Figure 7: Handoff sections of every task. The tasks from top to bottom are: *Square*, *Three Piece*, *Tool Hang*, *Coffee*, *Coffee Preparation*.

E Tasks

We describe the nine tasks in the main paper in more detail.

Square and **Square Broad**. The robot must pick up a nut and place it onto a peg. This task has 1 handoff section, where the learned agent places the nut. The **Broad** version increases the initialization range of both the nut and the peg.

Three Piece and **Three Piece Broad**. The robot must assemble a structure by inserting one piece into a base and placing another piece on top of the first. This task has 2 handoff sections, where the learned agent places the two pieces. The **Broad** version increases the initialization range of all three pieces including the base.

Tool Hang and **Tool Hang Broad**. The robot must first insert a L-shaped piece into a base to assemble a frame, then hang a wrench off of the frame. This task has 2 handoff sections, where the learned agent inserts the L-shaped piece and hangs the wrench. The **Broad** version increases the initialization range of all three pieces (base, L-shaped hook, and wrench).

Coffee and **Coffee Broad**. The robot must pick up a coffee pod, insert it into a coffee machine, and close the lid. This task has 1 handoff section where the learned agent inserts the pod and closes the lid. The **Broad** version increases the initialization range of the pod and the coffee machine.

Coffee Preparation. This is an extended version of **Coffee**. The robot must place a mug onto the coffee machine, open the lid, open the drawer where the coffee pod is placed, pick up the pod, insert the pod into the coffee machine, and finally close the lid. This task has 3 handoff sections where the learned agent (1) places the mug and opens the lid, (2) opens the drawer, and (3) inserts the pod and closes the lid.

See Figure 7 for an illustration of all the handoff sections.

Table 3: Mean and standard deviation (in parenthesis) of success rates out of 5 seeds.

	BC	**RL** [15]	**Ours**
Square	92.4 (5.5)	83.6 (36.7)	99.2 (1.8)
Square Broad	96.4 (4.1)	100.0 (0.0)	96.4 (5.4)
Coffee	96.8 (4.1)	40.0 (52.1)	88.0 (26.8)
Coffee Broad	41.6 (6.7)	23.2 (12.1)	84.4 (8.3)
Three Piece	63.6 (6.7)	0.0 (0.0)	84.0 (34.7)
Three Piece Broad	25.2 (7.7)	0.0 (0.0)	78.4 (5.0)
Tool Hang	9.2 (4.6)	0.0 (0.0)	54.0 (46.8)

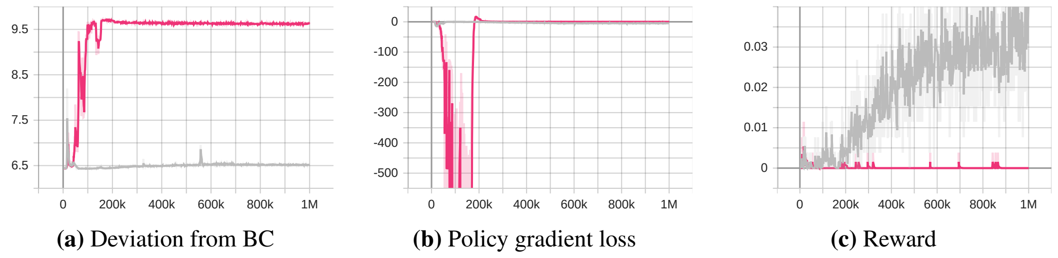


Figure 8: Comparing the (a) *Deviation from BC*, (b) *policy gradient loss*, and (c) *reward* training curves of a successful run (marked as grey) and a failed run (marked as red) in *Tool Hang*.

F Variance Across Seeds

In Figure 3, we show the best run out of 5 seeds. Here we provide the mean and standard deviation of the success rates in Table 3. We observe that although SPIRE still outperforms BC in terms of mean success rate in most of the tasks, our method exhibits unusually high variances in some of the tasks, for example, *Coffee*, *Three Piece*, and *Tool Hang*. In those tasks, one or more runs result in a performance significantly lower than the rest. Specifically,

- In *Coffee*, one run has 40% success rate, while the rest are all 100%;
- In *Three Piece*, one run has 22% success rate, while the rest are at least 98%;
- In *Tool Hang*, one run has 0% success rate and one has 6%, while the rest are at least 82%.

Reinforcement learning methods are known to have high variances, especially in sparse reward settings. SPIRE partially alleviates this problem by enforcing the KL penalty for deviating from an anchor policy. However, in practice, such deviation can still happen.

Figure 8 compares the training curve of a successful run (with 88% final success rate) and a failed run (with 0% final success rate). The policy in the failed run drastically deviated from the BC policy early on in the training. This is likely related to the unusually large policy gradient loss, which the KL penalty term was unable to match and failed to constrain the policy.

In our experiments, such an abrupt decrease in policy gradient loss happens frequently, with varying scales and timing, causing the training results to have high variance. Using an adaptive weight of the KL penalty might be a potential solution, which we wish to investigate in future work.

We do not believe 5 seeds are enough to quantitatively reflect the chance of such sudden deviation happening. An alternative solution would be to compare only the results where such deviation did not happen, which is why we chose to report the top-1 performing seed in our main paper.