2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Constrained Bootstrapped Learning for Few-Shot Robot Skill Adaptation

A K M Nadimul Haque[1], Fouad Sukkar[1], Lukas Tanz[2], Marc G. Carmichael[1] and Teresa Vidal-Calleja[1]

***Abstract*— In this paper, we propose a robot skill-learning method that facilitates fast adaption to new tasks online. Our method is based on a hybrid learning from demonstration and reinforcement learning approach, which seeds learning with a compact and structured skill model, leading to efficient and stable behaviours. To facilitate fast skill adaption, we propose a bootstrapped learning framework that learns a policy for adapting a skill model across a wide range of initial conditions in simulation. This policy is then used to bootstrap a refinement process that quickly adapts the learnt skill model to new initial conditions in a few learning iterations. Our refined skill model is designed to be deployable on hardware and can correct for discrepancies between the simulation and the real world. Furthermore, we propose a novel method for constraining policy exploration to promising trajectories, which is crucial for enabling manipulation in complex environments. We evaluate our framework in simulation and hardware in multiple environments with varying task complexity. We showcase that compared to the state-of-the-art, which achieves an average success rate of only 56.6% across three different tasks of varying difficulty, our algorithm significantly outperforms it with an average success rate of 90%.**

I. INTRODUCTION

One of the core capabilities of an intelligent agent is the ability to adapt to different tasks. Equipping robots with such capability is still a challenge [1]. Recent work in this area has shown promise in simulated environments. However, it tends to perform poorly in the real world [2].

Recently, the idea of translating human skills to robots via hybrid approaches that combine deep reinforcement learning (RL) and learning from demonstration has been successful in producing robust policies on real hardware [3], [4]. A robot skill model is initialised from human demonstrations. To account for discrepancies, such as localisation errors, this model is then refined with RL through physical interaction with the world.

Key to their approach is maintaining a physically meaningful skill structure within the RL refinement process. This structure results in more efficient learning and stable behaviour, which is particularly important in real-world settings. One of the drawbacks of this method is that the learnt policy is trained for a specific task with static initial conditions. Thus, time-consuming training must be performed if the task changes.

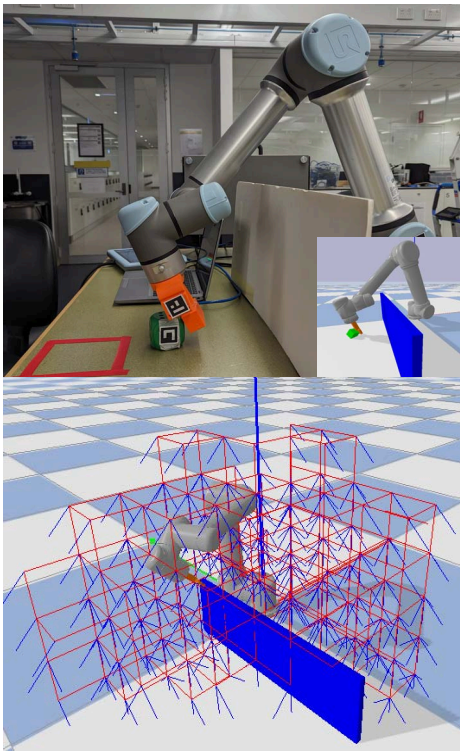

Fig. 1. The agent learns to tune provided demonstrations in a simulation constrained with feasible and efficient poses in the task space under varying initial conditions (bottom). The agent then refines the policy for the real-world skill in simulations before executing in the real world (top).

In this paper, we aim to overcome this limitation by enabling few-shot robot skill adaption. We adopt the hybrid approach; however, instead of training a refinement policy for a single task in hardware, we utilise simulation to learn a policy across a range of initial conditions. This policy is then used to bootstrap the refinement process for new initial conditions, resulting in much faster skill adaption. Our method is able to deal with discrepancies between simulation and hardware via a sensor feedback mechanism, which feeds into the RL policy learning (see Fig. 1).

In addition, we show that learning performance degrades in challenging environments where the robot must manipulate objects in close proximity to obstacles. To overcome this, we propose a novel method for constraining RL policy exploration to only consider promising trajectories during training. We evaluate our framework across three different

This work was supported by the Industrial Transformation Training Centre (ITTC) for Collaborative Robotics in Advanced Manufacturing (also known as the Australian Cobotics Centre) funded by ARC (Project ID: IC200100001)

[1] A K M Nadimul Haque, Fouad Sukkar, Marc Carmichael and Teresa Vidal-Calleja are with the Robotics Institute, University of Technology Sydney, Australia (e-mail: akmnadimul.haque@student.uts.edu.au, {fouad.sukkar,marc.carmichael,teresa.vidalcalleja}@uts.edu.au)

[2] Lukas Tanz is with the Institute for Machine Tools and Industrial Management, Technical University of Munich, Munich, Germany (e-mail: lukas.tanz@iwb.tum.de)

tasks of varying difficulty, with the strictest task involving complex orientation changes and a high chance of collisions. We show quantitative results both in simulation and in hardware, outperforming the state-of-the-art in both cases.

To summarise, the key contributions are as follows:

- A bootstrapping method for quickly adapting learnt skills to new initial conditions.
- A novel constrained policy exploration method for enabling manipulation in complex environments.
- A refinement process for hardware deployment, utilising sensor feedback to correct for sim-to-real discrepancies.

II. Related Work

Learning from demonstration (LfD) is a method for teaching robots skills via demonstration. Due to their robustness and fast reactivity to perturbations, LfD has been a prominent method in robotic applications. There are several methods for teaching, for example via teleoperation [5], [6], kinesthetic teaching [7], [8] and direct recording of human motion [9], [3]. Dynamic movement primitives (DMP) [10] and Gaussian Mixture Models (GMM) are common representations used to analytically describe robot motion over time. In this paper, we utilise the latter due to its compact parameterisation.

The main limitation of LfD methods is their inability to deal well with noisy perception. This is particularly important when the demonstration system, for example, a video recording of a human or a simulated model, differs from the target system where the policy is deployed. The work in [3], [4] aims to address this by complementing the GMM-parametrised dynamical system with RL. The RL agent learns to correct for sensor noise through physical interaction with the world.

However, the main drawback of their method is that costly training is required for every new task. Another limitation is that their performance degrades significantly in complex manipulation tasks. In contrast to their problem settings, we are interested in enabling manipulation tasks in close proximity to obstacles whilst being robust to sensor noise and fast to adapt to different tasks.

One way to mitigate the need for extensive re-training is through bootstrapped learning. The work in [11] proposed a normalised actor-critic algorithm that learns an initial policy from given demonstrations before refining it depending on the environment. In [12], a motion planner is used to bootstrap their model, where the motion planner finds successful, optimal trajectories that require further tuning to complete a task. A policy search method then improves on this to solve for the new environment. However, these methods either weren't applicable to real-world settings, or the tasks were simplistic and didn't require any complex manipulation. In contrast, our approach is designed to work with real hardware and is robust in complex manipulation tasks.

III. Problem Formulation

A. Background

1) SAC-GMM: Gaussian Mixture Models (GMM) are a probabilistic model that combines multiple Gaussians in order to model complex probability distributions. SAC-GMM [3] utilises GMMs as a compact model of robot skills in trajectory distribution space. Robot skills are provided as a set of demonstrations, which are then used to model the joint probability density of robot poses.

To deal with localisation errors, for example, due to differing demonstration and target systems, a deep RL algorithm called soft actor-critic (SAC) is utilised to learn to refine the compact parameters of the GMM through interaction with the world. This refinement is achieved by setting the actions output by the SAC agent to changes in the GMM parameters. The SAC agent optimises its policy for skill success.

Key to their approach is maintaining a physically meaningful structure within the RL refinement process. Additionally, learning efficiency is gained by reducing the action space of the RL agent through the use of a parameterised model.

B. Few-shot Robot Skill Adaption

SAC-GMM trains a refinement policy for a specific task with the same initial conditions (IC). Instead, we wish to adopt their hybrid approach but instead learn a refinement policy that can quickly adapt to a wide range of ICs.

We consider a robot skill as a sequence of motions that drives the robot from an initial state to a target state while carrying out a desired task, which may include manipulating the environment. We define such a motion as a trajectory of poses $\boldsymbol{T} = \{\boldsymbol{p}_1, ..., \boldsymbol{p}_n\}$, where $\boldsymbol{p_i} \in SE(3)$. During trajectory execution, the robot makes observations $\boldsymbol{O} = \{\boldsymbol{o}_1, ..., \boldsymbol{o}_n\}$ from the environment, such as distance and state environment.

We initialise such skills via a set of demonstrations $\mathcal{D}_s = \{\boldsymbol{T}_1, \boldsymbol{T}_2, ..., \boldsymbol{T}_m\}$. To aid with skill refinement, we wish to represent the distribution of trajectories with a compact parameterised model, $f_\theta(t) = \boldsymbol{p}$, where t is time, such that the number of the parameters is far less than the number poses in the trajectory.

Similar to [3], we wish to find a skill refinement policy $\pi_\phi(\boldsymbol{p}, \boldsymbol{o}) = \Delta\boldsymbol{\theta}$ that operates on the parameters of our trajectory model. However, in contrast to learning a new policy for each new initial condition, we aim for this policy to be able to adapt $f_\theta(t)$ with a relatively small number of learning steps or few-shot. Thus, we wish to learn a policy $\hat{\pi}_\phi$ that can effectively bootstrap the refinement process and result in fast skill adaption to any given IC.

IV. Proposed Framework

Here, we provide a detailed description of the few-shot robot skill adaption framework. An overview of the proposed framework is shown in Fig. 2.

A. Bootstrapped Learning

Learning $\hat{\pi}_\phi$ is hard because the range of ICs for a given environment is often large and, in the case of continuous domains, infinite. One could naively attempt to train on all possible ICs, however this would be intractable in terms of computation time. Thus, to overcome this intractability, we

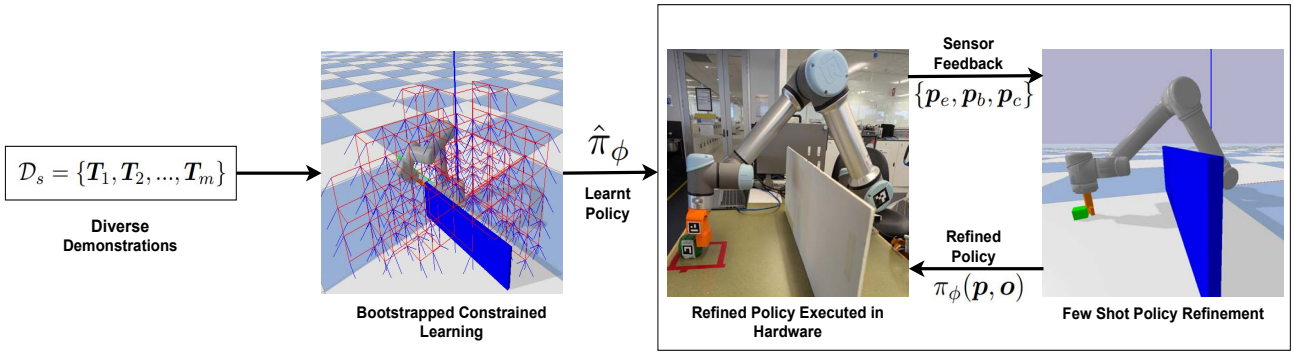

Fig. 2. Proposed framework: The agent learns in simulation from demonstrations parameterised with a GMM model under varying ICs, resulting in a bootstrapped policy that can be quickly refined and adapted to new environments, facilitating sim-to-real transfer.

propose to train $\hat{\pi}_\phi$ on a sampled sparse set of ICs and use this policy to bootstrap the refinement process.

While learning of $\hat{\pi}_\phi$ occurs in simulation, we facilitate transfer to real hardware. To account for discrepancies between the simulation and the real world, we train with additive Gaussian noise to the simulated observations. Furthermore, to account for localisation error in the hardware, we consider sensor feedback after a batch of time steps during refinement, in a similar fashion to [3]. To distinguish between a trajectory pose time step and a batch of time steps, we use $\hat{t}$ to denote the latter.

To help with invariance to ICs, we choose our observations to be distance vectors from the robot end effector to the closest obstacle, $\boldsymbol{x}_o$, and the target pose, $\boldsymbol{x}_g$. Thus our state observations $\boldsymbol{s}_{\hat{t}}$ and actions $\boldsymbol{a}_{\hat{t}}$ at each time step batch are:

$$\boldsymbol{s}_{\hat{t}} = \{\boldsymbol{x}_o, \boldsymbol{x}_g\}; \boldsymbol{a}_{\hat{t}} = \Delta\boldsymbol{\theta}\,. \tag{1}$$

Thus, the optimal policy is

$$\hat{\pi}_\phi = \arg\max_{\pi} \tau \sim \pi \sum_{\hat{t}=0}^{\nu-1} \gamma^{\hat{t}} \Big(\boldsymbol{r}(\boldsymbol{s}_{\hat{t}}, \boldsymbol{a}_{\hat{t}}, \boldsymbol{s}_{\hat{t}+1}) + \beta H\left(\pi(\cdot|\boldsymbol{s}_{\hat{t}})\right) \Big) \tag{2}$$

where r is the obtained reward of that state-action pair τ, ν is the number of batch time steps in the trajectory, γ is the discount factor, and β is the temperature parameter weighting the entropy $H(\pi(\cdot|\boldsymbol{s}_{\hat{t}}))$. Entropy is estimated as the expected value of the negative log probability, averaging over every possible action by the policy in a given state.

$$H(\pi(\cdot|\boldsymbol{s}_{\hat{t}})) = -E_{\boldsymbol{a}_{\hat{t}} \sim \pi(\cdot|\boldsymbol{s}_{\hat{t}})} \left[\log \pi(\boldsymbol{a}_{\hat{t}}|\boldsymbol{s}_{\hat{t}})\right] \tag{3}$$

Our training process is as follows. Given an environment, we generate a set of demonstrations for a sparse set of ICs, denoted $\mathcal{D}_s$. We parameterise $f_\theta(t)$ as a GMM and estimate the joint probability density $\mathcal{P}(\boldsymbol{p}, t)$ using $\mathcal{D}_s$. Using a modified version of SAC-GMM, we train $\hat{\pi}_\phi$ to adapt parameters $\boldsymbol{\theta} = \{\boldsymbol{\alpha}_k, \boldsymbol{\mu}_k\}_{k=1}^K$, where $\boldsymbol{\alpha}_k$ are mixing weights and $\boldsymbol{\mu}_k$ are means for K Gaussians, for a pre-set number of episodes across a range of randomly sampled ICs.

Each episode of our modified SAC-GMM runs as follows. We sample random initial ICs. The parameters of $f_\theta(t)$ are updated with $\hat{\pi}_\phi$ given observations $\boldsymbol{O}$. Given the updated $f_{\theta+\Delta\theta}(t)$ we use Gaussian mixture regression (GMR) to produce $\boldsymbol{p}$ given t as the conditional distribution $\mathcal{P}(\boldsymbol{p}|t)$. Then after N interaction time steps a new $\boldsymbol{s}_{\hat{t}}$ and reward $r_{\hat{t}}$ are produced. This process is repeated for a set number of time steps or until collision with an obstacle occurs.

The SAC agent optimises $\hat{\pi}_\phi$ for skill success by maximising the total reward for each episode. Note that the GMM parameters are reset to the base $f_\theta(t)$ after each episode. The result is a policy that can adapt a base trajectory model to a wide range of ICs.

B. Constrained Policy Exploration

The bootstrapped learning process can be impractically slow in challenging environments. For example, a high-dimensional robot arm can achieve the same end-effector pose via many joint configurations. Due to this, particularly when moving around obstacles, pose trajectories that are seemingly short in task space can lead to long and non-smooth configuration trajectories. This is problematic for RL training since the agent needs to explore a large number of poses around these problematic regions in the environment before determining what is good behaviour.

To mitigate this issue, we utilise the Hausdroff approximation planner (HAP) [13]. HAP decomposes a user-defined task space into subspaces such that the path between any two points close in a particular subspace maps to a short, smooth and collision-free configuration space trajectory. HAP can output multiple subspaces for a given task space; thus, the one with the largest task space coverage is used. This subspace is represented by a discrete roadmap of poses (see Fig. 1).

Thus, using this subspace knowledge, we indirectly constrain the explorable task space by the RL agent. We achieve this by checking that every pose $\boldsymbol{p}$ predicted by $f_{\theta+\Delta\theta}(t)$ at time step t is within a set distance from any discrete subspace pose before continuing the training episode. Otherwise, we terminate the episode. Thus, we achieve efficient RL training by focusing exploration towards promising trajectories.

C. Fast Online Policy Refinement

Given a new IC in an online scenario, we initialise the refinement policy to the learnt $\hat{\pi}_\phi$ and set our initial trajectory

model to $f_\theta(t)$. In order to compensate for any sensor noise, localisation errors, and bridge any sim-to-real gap, we propose a further few-shot refinement of the policy.

We first update the simulation to reflect the real world using sensor feedback. This includes the pose of end-effector $\boldsymbol{p}_e$, robot base $\boldsymbol{p}_b$ and the cube $\boldsymbol{p}_c$. The policy is then trained, using the same objective in Eq. 2, in the updated simulation for a small number of steps to obtain the adapted policy π_ϕ and consequent $f_{\theta+\Delta\theta}(t)$. $f_{\theta+\Delta\theta}(t)$ then produces the pose batch $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\}$ which is then executed on the hardware. This process is repeated for ν batch time steps with π_ϕ and $f_{\theta+\Delta\theta}(t)$ from the previous iteration and the new sensor information post-execution.

V. Experiments

In this section, we validate the proposed approach. First, we describe the experimental design, including the environment setup and evaluation method. Then, we present comparative results and a discussion that provides insights into the benefits of our framework.

A. Experimental Design

In our experiments, we first validate our framework using a baseline task. To highlight the benefits of our framework, we then provide an evaluation of two scenarios that involve manipulating an object in close proximity to obstacles.

We validate our results both in simulation via Pybullet and on a real UR5 manipulator. We further design an experiment to determine the effects of policy bootstrapping.

For transferring to hardware, we use ARUCO markers [14] to obtain the real-world state information and adjust the simulator to match the real world. We place these markers (tags) on the end-effector, the obstacle, and the object to be manipulated to obtain the distance from the camera. We also place a tag on the robot base to recover the transformation from the camera to the robot base to obtain the required distances in the robot frame.

B. Scenarios

1) Drawer pulling: Initially, we consider a drawer-pulling scenario as shown in Fig. 3 (left), where the drawer spawns in different positions. For training, the agent is given 30 demonstrations with different start and end poses. The cabinet positions vary as well.

2) Cube Pushing - Simple: This is a more challenging scenario where the arm must carefully avoid collision with a wall obstacle. In addition, the arm must manipulate a cube by pushing it to a specified goal position.

3) Cube Pushing - Difficult: To make a more compelling case, we design a more difficult scenario where the cube spawns much closer to the obstacle (see Fig. 3 (right)). This requires re-orientation of the end effector to avoid collision with the wall. Thus, there is a much smaller set of valid poses the arm can explore around the spawn locations of the cube. This experiment highlights the benefit of the subspace knowledge provided by HAP.

C. Benchmarks

We benchmark our method against four different approaches.

1) Baseline SAC: We use the baseline SAC algorithm from [15]. Inputs to SAC are the current position of the end-effector, the distance from the obstacle and the goal as the state inputs.

2) GMM: We regress a trajectory using the GMM fit to the demonstrated trajectories, i.e. the base $f_\theta(t)$.

3) SAC-GMM: We implement the SAC-GMM similar to [3].

4) SAC-GMM with baseline poses: We supplement SAC-GMM with a constrained task space. However, these are simply all valid IK solutions and not necessarily those that have short paths in both task space and configuration space. This task space is used as input for the HAP algorithm.

D. Simulation Results

We present the results in simulations of each framework. In each case, the agent is provided with the GMM used in the training. Each experiment consists of 20 runs with differing initial conditions, i.e. initial start and goal poses. The results of the simulations are shown in Fig. 4 and Table I.

Baseline SAC: Our results show that the baseline SAC algorithm cannot produce any meaningful trajectory, even when aided by the demonstrations.

GMM: The base GMM $f_\theta(t)$ produces a trajectory that is approximately centred around the demonstrated trajectories. Thus, it succeeds only when, by chance, the goal is near the average of those in the provided demonstrations. ***SAC-GMM***: Overall, SAC-GMM performs slightly better than GMM, with success rates of 60% and 70% in drawer-pulling and easier cube-pushing tasks, respectively. However, it drastically underperforms compared to the other algorithms in the strict scenario, with only a 40% success rate.

SAC-GMM with baseline poses: SAC-GMM aided with baseline poses performs well in the drawer-pulling scenario with 100% accuracy. In the case of the pushing tasks, it performs better than SAC-GMM, with success rates of 75% and 55% for the easy and stricter cases, respectively.

SAC-GMM-HAP (Ours): Our proposed algorithm achieved 100% accuracy in the drawer-pulling task, similar to SAC-GMM with baseline poses. However, in our easier cube-pushing task, which involves close proximity manipulation around an obstacle, the benefits of the HAP subspace are exemplified since the manipulator must carefully consider its orientation when approaching the wall obstacle. Our framework significantly outperforms the others with a success rate of 95%. In the case of the cube-pushing task with strict demonstrations, our framework achieves a 75% success rate, considerably more than the others.

E. Hardware Results

We observe similar results in the hardware experiments, which validate our sensor feedback mechanism's ability to correct for sim-to-real discrepancies. Fig. 5 shows images of the experimental setup. The comparative success rates are shown in Table II.

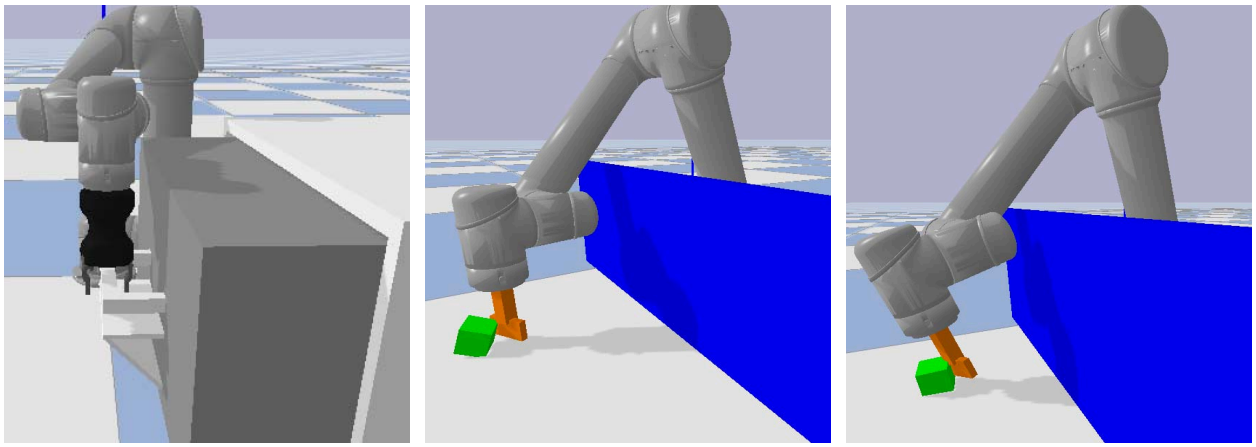

Fig. 3. Simulated experiments with increasing difficulties. Left - Drawer pulling task involving no change in orientation, Middle - Cube pushing task with some changes in orientations and relatively safe operating zone, Right - Strict cube pushing tasks involving complex orientation changes and a high chance of collision both with the plane and the obstacle.

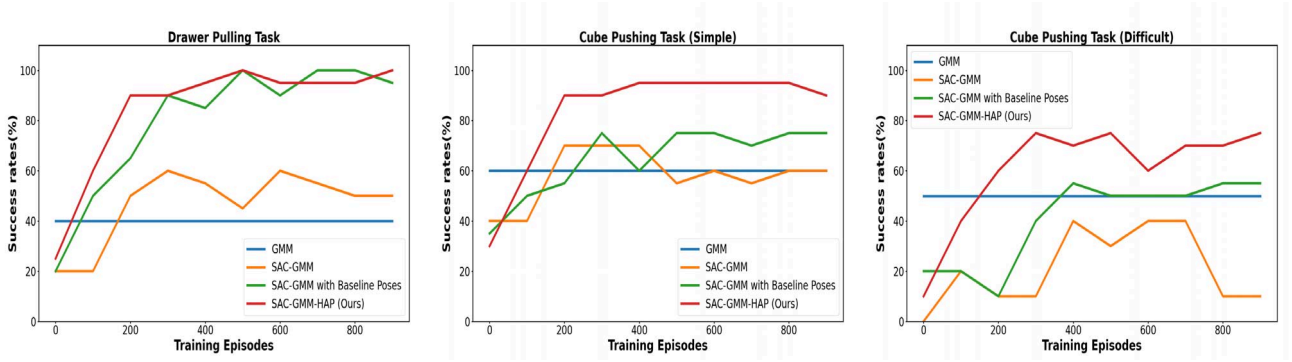

Fig. 4. Success rates in simulations over training episodes across the three tasks. In all the tasks, our framework outperforms others, achieving high success rates when tested in varying ICs.

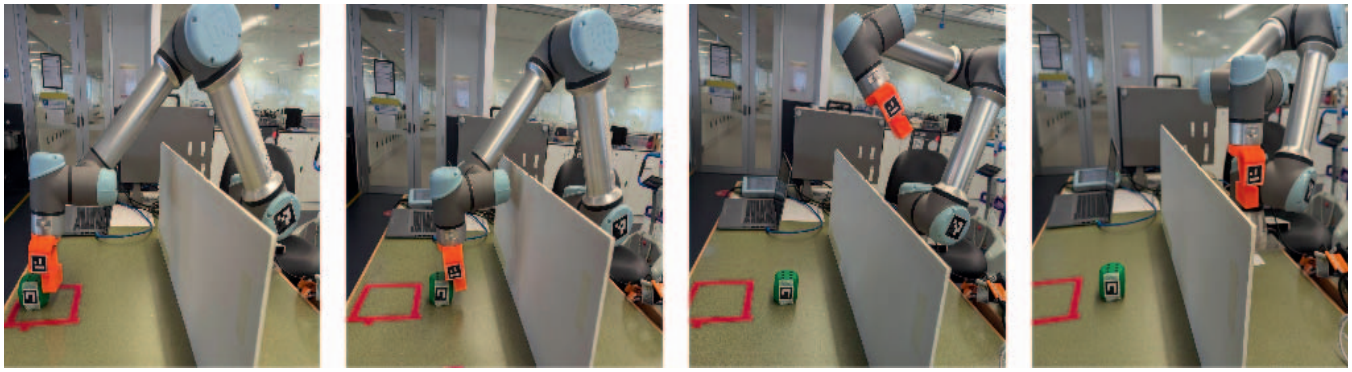

Fig. 5. Bootstrapped policy executed on hardware with the UR5 robot.

TABLE I

SUCCESS RATES ACROSS EXPERIMENTS IN SIMULATION

Task	GMM	SAC-GMM	SAC-GMM with Baseline Poses	**SAC-GMM-HAP (Ours)**
Drawer Pulling	40%	60%	100%	**100%**
Cube Pushing (Simple)	60%	70%	75%	**95%**
Cube Pushing (Difficult)	50%	40%	55%	**75%**

TABLE II

SUCCESS RATES FOR CUBE PUSHING (DIFFICULT) TASK IN HARDWARE

GMM	SAC-GMM	SAC-GMM with Baseline Poses	**SAC-GMM-HAP (Ours)**
40%	10%	40%	**70%**

F. Discussions

Our results indicate that learning is required to adapt the base GMM model in order to reliably carry out the demonstrated tasks successfully. We observe that the baseline SAC-GMM cannot reliably respond to changes in ICs and, in some cases, performs worse than the base GMM model due to unstable training. We attribute this to the fact that a large amount of training effort is required to navigate around the complex obstacle configurations. This leads to SAC's increasing exploration of solutions well outside the demonstrated trajectory space and, thus, a collapse of the base model.

As our results show, constraining the RL policy search space significantly improves the performance in terms of robustness to different ICs. The policy performs substantially better even when constraining just through the baseline poses. The imposed restriction guides the exploration towards trajectories within the manipulator's task space. Thus, SAC samples instances that lead to meaningful trajectories that the manipulator can execute. This is further improved with the help of HAP. Since HAP provides a subspace where trajectories are efficient, RL exploration is focused on promising regions of the workspace, greatly increasing training speed and success rates compared to benchmark methods.

VI. CONCLUSION

This paper presents a novel constrained bootstrapped learning approach for few-shot robot skill adaptation. We adopt a hybrid learning from demonstration and reinforcement learning approach, which can deal with varying initial conditions and model discrepancies, such as localisation errors. We showed that with bootstrapped learning, our method performs significantly better in this setting with a small number of training episodes compared to the state-of-the-art. This has promising prospects for real-world robotic applications where adaptability and quick learning are paramount. Furthermore, we demonstrate the necessity of our novel constrained policy exploration in complex manipulation tasks.

REFERENCES

[1] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual review of control, robotics, and autonomous systems*, vol. 3, pp. 297–330, 2020.

[2] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," *arXiv preprint arXiv:1511.06295*, 2015.

[3] I. Nematollahi, E. Rosete-Beas, A. Röfer, T. Welschehold, A. Valada, and W. Burgard, "Robot skill adaptation via soft actor-critic gaussian mixture models," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 8651–8657, IEEE, 2022.

[4] H. Ziesche and L. Rozo, "Wasserstein gradient flows for optimizing gaussian mixture policies," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[5] O. Kilinc and G. Montana, "Reinforcement learning for robotic manipulation using simulated locomotion demonstrations," *Machine Learning*, pp. 1–22, 2022.

[6] L. Wang, Y. Ling, Z. Yuan, M. Shridhar, C. Bao, Y. Qin, B. Wang, H. Xu, and X. Wang, "Gensim: Generating robotic simulation tasks via large language models," *arXiv preprint arXiv:2310.01361*, 2023.

[7] G. Ajaykumar, M. Stiber, and C.-M. Huang, "Designing user-centric programming aids for kinesthetic teaching of collaborative robots," *Robotics and Autonomous Systems*, vol. 145, p. 103845, 2021.

[8] P. Aliasghari, M. Ghafurian, C. L. Nehaniv, and K. Dautenhahn, "Kinesthetic teaching of a robot over multiple sessions: Impacts on speed and success," in *International Conference on Social Robotics*, pp. 160–170, Springer, 2022.

[9] C.-J. Liang, V. R. Kamat, and C. C. Menassa, "Teaching robots to perform quasi-repetitive construction tasks through human demonstration," *Automation in Construction*, vol. 120, p. 103370, 2020.

[10] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *The International Journal of Robotics Research*, vol. 42, no. 13, pp. 1133–1184, 2023.

[11] Y. Gao, H. Xu, J. Lin, F. Yu, S. Levine, and T. Darrell, "Reinforcement learning from imperfect demonstrations," *arXiv preprint arXiv:1802.05313*, 2018.

[12] B. Abbatematteo, E. Rosen, S. Tellex, and G. Konidaris, "Bootstrapping motor skill learning with motion planning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4926–4933, IEEE, 2021.

[13] F. Sukkar, J. Wakulicz, K. M. B. Lee, and R. Fitch, "Motion planning in task space with gromov-hausdorff approximations," *arXiv preprint arXiv:2209.04800*, 2022.

[14] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[15] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.