Deep Skill Chaining from Incomplete Demonstrations

Minjae Kang and Songhwai Oh

***Abstract*— In general, humans can easily assign a sequence of simple tasks for performing complex tasks. If a person gives an agent an order of simple tasks to carry out a complex task, we can find a skill sequence efficiently by learning the corresponding skills. However, independently trained low-level skills (simple tasks) are incompatible, so they cannot be performed in sequence without additional refinement. In this context, we propose a method to create a skill chain by connecting independently learned skills. In the experiment, we perform manipulation tasks with RGBD images as input in the Baxter simulator implemented using MuJoCo. We verify that skill chains can be successfully trained from incomplete data while confirming that the agent can be trained much more efficiently and stably than baselines.**

I. INTRODUCTION

In general, a person develops a universal plan at high-level when performing complex tasks. If a person gives an agent a simple-task order to carry out a complex task, we can find a skill sequence efficiently by learning the corresponding skills. In this paper, we propose a method to solve complex tasks by combining a rough skill sequence with the study of skill chaining [1], [2]. Our algorithm allows us to learn the entire skill chain without intact complex-task demonstrations based on a given skill sequence. To avoid confusion, a task that can be performed with a single skill is defined as a *simple task*, and in contrast, a task that can only be solved with multiple simple skills is called a *complex task*.

We define two types of skills: the *base skill* learned from demonstrations and the *bridge skill* connecting two consecutive base skills. Base skills can be trained simply through imitation learning. However, it is challenging to collect data for bridge skills because they do not perform specific tasks compared to base skills. Consequently, we use reinforcement learning to train bridge skills. Learning bridge skills requires solving two issues: 1) Demonstrations for training bridge skills are not available, and only sparse rewards exist. 2) It is difficult to know whether bridge skills operate properly and at what point to switch to the next skill. First, we propose a latent-distance reward function to complement the deficient reward. Fragmented demonstrations are labeled with simple tasks, so states can be mapped to a latent space to form clusters using a neural network named a *clustering network*. In this latent space, we define a novel reward function using the Euclidean distance between the current latent vector and the target cluster center, making bridge skill training more efficient. Next, the skill switching issue is addressed using a binary classifier named a *skill switch*. It makes the skill be connected by determining whether the current state is a state capable of performing the skill that follows.

M. Kang and S. Oh are with the Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul 08826, Korea (e-mail: minjae.kang@rllab.snu.ac.kr, songhwai@snu.ac.kr). This work was supported by Institute of Information Communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-01190, [SW Star Lab] Robot Learning: Efficient, Safe, and Socially-Acceptable Machine Learning).

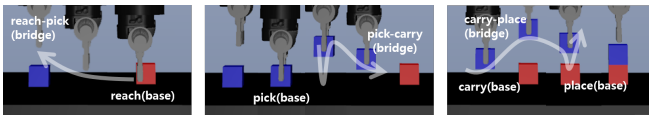


Fig. 1: Skill chain trajectory for performing *stack*.

In the experiment, we perform three manipulation tasks with two RGBD images as input in the Baxter simulator implemented on MuJoCo. We successfully obtain the skill chains consisting of base and bridge skills that can perform complex tasks. Figure 1 shows a trajectory of the skill chain for *stack* found in this experiment. Furthermore, the proposed latent-distance reward function makes learning more reliable with higher performance compared to the sparse rewards. Our algorithm using latent-distance rewards shows success rates close to 0.8 for all tasks, while success rates of the skill chain trained with sparse rewards are lower than 0.6.

II. BACKGROUND

A. Option Framework

In the option framework [1], an option $\mathcal{O}$, which also considers a high-level action, has three components: 1) Policy $\pi_{\mathcal{O}} : \mathcal{S} \rightarrow \mathcal{A}$ for determining low-level action a from the current state s, 2) Termination condition $\beta_{\mathcal{O}} : \mathcal{S} \rightarrow \{0, 1\}$ for determining whether to maintain the current option $\mathcal{O}$, and 3) Initiation condition $I_{\mathcal{O}} : \mathcal{S} \rightarrow \{0, 1\}$ for determining which option is feasible from the given state s. Note that if the termination condition of the preceding skill is sufficient for the initiation condition of the skill followed, these two skills can always form a chain.

B. Reinforcement Learning

Assuming an Markov decision process (MDP), one of the representative methods for finding a optimal policy π^* is reinforcement learning (RL) [3], [4], [5]. In RL, we select the appropriate action a from state s by predicting state-action value, which indicates the expected reward sum that can be obtained after taking action a in the state s. The value function is determined by solving the Bellman equation as follows: $V_\pi(s) = \sum_a \pi(a|s) Q_\pi(s, a)$ and $Q_\pi(s, a) = \mathbf{r}(s, a) + \gamma \sum_{s'} V_\pi(s') T(s'|s, a)$, where $V_\pi(s)$ is a state value function that estimates expected cumulative reward when following policy π from state s, and $Q_\pi(s, a)$ is a state-action value over policy π.

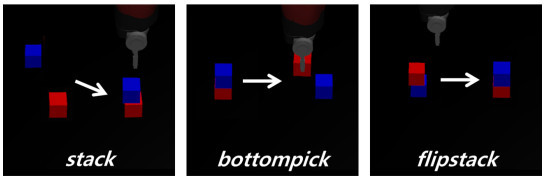
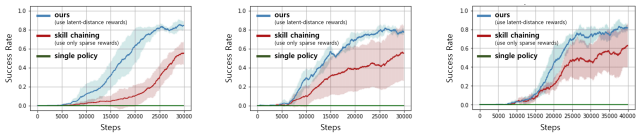


Fig. 2: This illustration shows initial and goal state examples of three manipulation tasks: *stack*, *bottompick*, and *flipstack*.



(a) Result of *stack* (b) Result of *bottompick* (c) Result of *flipstack*

Fig. 3: Success rate of *stack*, *bottompick*, and *flipstack* task. All graphs indicate the average of 100 consecutive episodes with five different random seeds.

III. METHOD

This section details how to discover a skill chain from incomplete demonstrations. To find a skill chain for complex tasks efficiently, we assume that we know the foundational execution sequence of simple skills that a human can suggest to accomplishing a particular task. For example, to perform pick-and-place, we first pick up a block and then place it on a fixed position. Base on a given skill sequence, we can define skills as two types: base skills corresponding to each element of a given sequence and bridge skills that connect two incompatible base skills. Finally, we form a skill chain that performs base skills and bridge skills alternatively.

To successfully connect base skills using a bridge skill, the bridge skill must have a termination condition β and an inner policy π. First, we use a binary classifier θ as skill switch to verify that the current state meets the termination condition. We train θ using cross-entropy loss $\mathcal{L}_\theta = -\sum_i [t_i \log(\theta(s_i)) + (1-t_i)\log(1-\theta(s_i))]$, where t_i indicates success or not when starting the following skill in state s_i. Second, the inner policy π is trained through reinforcement learning. For efficient training, we propose novel rewards named *latent-distance rewards*. Using the encoder of variational auto-encoder (VAE) ϕ and clustering network ψ, we convert continuous states s to the latent variables l to form a cluster according to each simple task. In this latent space, we define the latent-distance reward function for the bridge skill $\mathcal{O}_m$ that is implemented just before the base skill $\mathcal{O}_n$ as follows:

$$r_m(s_t, s_{t+1}) = \| \psi(\phi(s_t)) - c_i \|_2 - \| \psi(\phi(s_{t+1})) - c_i \|_2,$$

where i is the task index and c indicates the cluster center of $\mathcal{O}_n$. This reward function significantly reduces the searching space, allowing the agent to be successfully trained with only small number of steps.

IV. EXPERIMENT

In this section, we perform manipulation tasks using blocks in the Baxter simulation. There are four kinds of base skills that deal with blocks: *reach*, *pick*, *carry*, and *place*. In this experiment, we define three complex tasks: *stack*, *bottompick*, and *flipstack* as shown in Figure 2. Note that these three manipulation tasks use the same kind of base skills but perform different tasks in different skill order.

As shown in Figure 3, we verify how efficiently complex tasks can be solved from incomplete demonstrations using latent-distance rewards and skill switches. To find out the effectiveness of latent-distance rewards, we compare the performance of our proposed algorithm, which is trained using latent-distance rewards, and the baseline that trained using only sparse rewards. In addition, to show that our task is complex, we attempt to train with just a single policy.

First, the performance of the single policy indicates that *stack*, *bottompick*, and *flipstack* are complex tasks that are difficult to train with only one policy. Also, we can confirm that our proposed algorithm is faster and shows higher performance than the skill chain trained with only sparse rewards. This result indicates that training through latent-distance rewards makes exploration more efficient and leads to more stable learning. Furthermore, our experimental results show that as the number of skills executed accumulates, the performance gap with baseline increases. This is because the skill progresses in order, so the faster the preceding skill is trained, the more opportunities there are to train the skills that follow. Therefore, the more complex tasks that require longer skill sequences, the performance can be more improved as the effect of latent-distance reward function accumulates.

V. CONCLUSIONS

In this paper, we propose a new method that discovers the skill chain from incomplete demonstration. Using a foundational sequence of simple tasks proposed by humans, we can efficiently discover a skill chain for a complex task consisting of base skills and bridge skills. Moreover, latent-distance rewards are suggested for efficient training of abstract-purpose bridge skills. We use binary classifiers as skill switches to determine when to exit the bridge skill and start the next base skill. Finally, in the experiment, we performed manipulation tasks in the Baxter simulator, and we show that our proposed algorithm solves complex tasks efficiently and stably.

REFERENCES

[1] G. Konidaris and A. Barto, "Skill discovery in continuous reinforcement learning domains using skill chaining," *Advances in neural information processing systems*, vol. 22, pp. 1015–1023, 2009.

[2] A. Bagaria and G. Konidaris, "Option discovery using deep skill chaining," in *International Conference on Learning Representations*, 2019.

[3] I. Osband, C. Blundell, A. Pritzel, and B. V. Roy, "Deep exploration via bootstrapped DQN," in *Advances in neural information processing systems*, Dec 2016, pp. 4026–4034.

[4] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, 2017.