Robot Generating Data for Learning Generalizable Visual Robotic Manipulation

Yunfei Li[1,2*], Ying Yuan[1*], Jingzhi Cui[1], Haoran Huan[1], Wei Fu[1], Jiaxuan Gao[1], Zekai Xu[3], Yi Wu[1,2,†]

***Abstract*— It has been a popular trend in AI to pretrain foundation models on massive data. However, collecting sufficient offline training trajectories for robot learning is particularly expensive since valid control actions are required. Therefore, most existing robotic datasets are collected from human experts. We tackle such a data collection issue with a new framework called "robot self-teaching", which asks the robot to self-generate effective training data instead of relying on human demonstrators. Our key idea is to train a separate data-generation policy operating on the state space to automatically generate meaningful actions and trajectories with ever-growing complexities. Then, these generated data can be further used to train a visual policy with strong compositional generalization capabilities. We validate our framework in two visual manipulation testbeds, including a multi-object stacking domain and a popular RL benchmark "Franka kitchen". Experiments show that the final visual policy trained on self-generated data can accomplish novel testing goals that require long-horizon robot executions. Project website `https://sites.google.com/view/robot-self-teaching`.**

I. Introduction

Pretraining has recently become the most popular training paradigm in artificial intelligence. Large-scale training on a vast quantity of pre-collected data has demonstrated prominent generalization power in a wide range of downstream tasks, such as pattern recognition [1], image/video synthesis [2], and language understanding [3], [4], [5], resulting in a promising direction toward artificial general intelligence [6]. Similar efforts have been made in learning general robot control policies from a pre-collected large dataset of diverse robot trajectories [7], [8], [9]. Such a paradigm that is often called *offline reinforcement learning* (RL) can produce strong policies with minimal fine-tuning efforts [10], [9] or even in a zero-shot manner [11], [12].

A fundamental challenge for offline RL methods is how to collect sufficient robot trajectories. Robot trajectories are more expensive to collect than images or texts, since they should contain *the control actions* in addition to the robot states. The trajectories are typically collected from a heavily engineered robot system with human feedback [13] or from a human demonstrator directly teleoperating the robots [14], [15]. To overcome this challenge, some very recent works leverage pretrained video or text models for high-level behavior guidance and then additionally train an action generation model for control [16]. However, how to precisely ground the robot states and actions to the inputs of foundation models remains an open challenge.

In this paper, we tackle the data collection challenge for robot pretraining by *asking the robot to self-generate meaningful data*. Our key idea is to train a separate data-generation policy to automatically produce robot trajectories with ever-growing complexities. Then, these generated trajectories can be used for training the desired visual control policy. We call this framework "robot self-teaching" (RST) since all the pre-training trajectories are produced by the robot itself with minimal human intervention. We realize the RST framework in the domain of goal-conditioned multi-object manipulation, where a goal image specifies the goal state for the robot to reach. As illustrated in Fig. 1, we first collect a small-scale seeding dataset containing demonstrations of basic tasks such as manipulating a single object to warm-start the data-generation policy. Then the data-generation policy operating on the state space repeatedly discovers novel tasks with increasing difficulties and produces feasible solution trajectories to keep augmenting the dataset. Finally, this enhanced dataset is used to train our desired visual policy for strong compositional generalization over unseen tasks at test time in a zero-shot fashion.

The most critical challenge of our RST framework is to consistently discover meaningful novel tasks. This is nontrivial because the newly discovered task must be solvable for the data-generation policy so that valid control actions can be generated. Meanwhile, the discovered task cannot be too easy either. For example, the robot can always push a single object to random positions on a table to generate arbitrarily many valid trajectories. However, these trajectories may not provide effective training signals for the final visual policy. We propose a novel approach called *task expansion*, which progressively discovers novel and solvable tasks in an open-ended fashion. Task expansion takes the value function of the data-generation policy as a progression metric. Once the data-generation policy successfully reaches a desired goal state from an initial state, task expansion performs a state-space search to find a novel goal state such that it is reachable, i.e., a high value from the current state, but also sufficiently challenging, i.e., a low value from the initial state, so that the skill level of the data-generation policy can be progressively improved. Such a task expansion process naturally yields an open-ended task curriculum with paired control actions of ever-growing complexities.

We empirically validate the robot self-teaching framework in two testbeds. In a block-stacking domain with multiple long cuboids, our method gradually generates building

*Equal contribution
†Corresponding author `jxwuyi@gmail.com`
[1]Tsinghua University, Beijing, China `liyf20@mails.tsinghua.edu.cn`
[2]Shanghai Qi Zhi Institute, Shanghai, China
[3]Shanghai Jiao Tong University, Shanghai, China

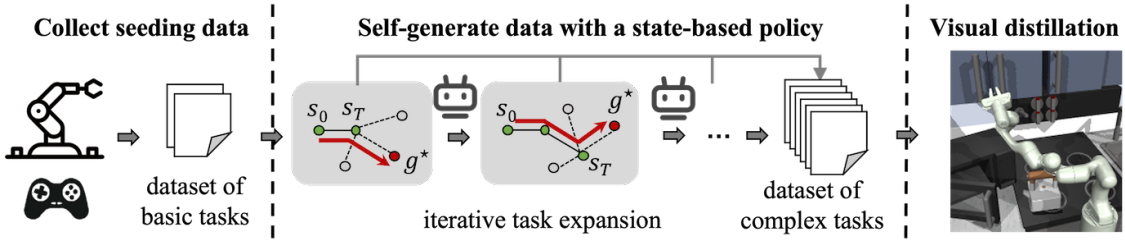


Fig. 1: Overview of the robot self-teaching framework. Starting from a small dataset of basic tasks collected by humans, the robot progressively generates a large dataset of complex behaviors via multiple rounds of task expansion. A visual policy is distilled from the generated dataset and could generalize to complex multi-stage tasks in a zero-shot manner.

structures from an initial dataset consisting of single block movement. The final visual policy achieves over 40% zero-shot success rate when tested on human-designed novel goal structures. We also evaluate RST in a popular offline RL benchmark "Franka kitchen". RST reaches long-horizon goals that require interacting with four components in the kitchen, utilizing seeding trajectories of single-object manipulation, while a planning-based offline RL baseline completely fails to solve these complex tasks.

II. Related Work

Robot learning with massive offline data: Offline RL studies the training of an RL agent from a fixed dataset [7], [17], [18], [19], and optionally fine-tuning the agent with some online interactions in target domains [10], [20].

Learning RL agents from offline datasets that could generalize to unseen and potentially long-horizon compositional tasks has attracted much research interest [14], [11], [12]. A notable line of work learns goal-conditioned agents from a dataset containing rich and diverse behaviors and could generalize to long-horizon tasks by stitching policies across different episodes [11]. Some recent works explicitly train a planner from offline datasets that can compose goal-conditioned policies [14], [21], [22] or extract a hierarchy of policies [23] to better deal with temporally extended problems. Our method similarly starts from an offline dataset, but instead of only learning from the provided data, our agent procedurally creates tasks and solutions with a higher level of compositionally by itself to greatly enhance the dataset, which is an under-explored direction in offline RL.

There is also success in leveraging the reasoning ability of large models to scale up robot policies learned from offline datasets to long-horizon tasks [24], [25], [26]. They bypass the challenging part of learning complex policies with external pre-trained models and the robot only focuses on basic tasks. In contrast, our agent self-augments data of basic tasks into a massive dataset containing complex behaviors and distills a flat policy that itself is capable of multi-stage problems. A recent work BOSS [27] leverages LLM to chain language-conditioned tasks into more complex ones captioned with natural language, while we focus on visual manipulation tasks that require more precise goal specification such as stacking.

Curriculum generation: A bunch of literature in curriculum learning [28] (CL) for RL studies how to create a curriculum of subgoals/initial states to accelerate the convergence to the most challenging tasks [29], [30], [31], [32]. These methods propose to sample tasks with moderate difficulty for the current agent and result in a curriculum of tasks from easy to hard. Task expansion is technically similar to goal-generation methods, but with very distinct settings. CL typically assumes the prior knowledge about the desired tasks to solve, while our agent does not know the existence of any targeted tasks in a prior and generates increasingly more complex tasks in an open-ended manner. Our open-ended generation of tasks and their solutions is conceptually similar to evolutionary environment generation [33], [34]. Similar ideas of creating tasks with an ever-growing complexity have also been explored in other domains such as exploration [35] and representation learning [36].

Teacher-student learning: Our framework leverages the output of a state-based policy ("teacher") to distill a visual policy ("student"), which is teach-student learning or privileged learning [37] commonly adopted in robot learning. A teacher policy is first trained with privileged information such as ground truth states and even unobservable environmental factors, then distilled to a student policy with the available sensory inputs in deployment. Teacher-student learning has been applied to mitigate the optimization challenges in robot control [38], [39], [40] and for sim-to-real transfer [41]. RST runs task expansion using state information to more easily discover meaningful new tasks, and it leverages the rollouts of state-based policies to teach a visual policy, which does not require privileged information during test time.

III. Preliminary

We consider the setting of goal-conditioned Markov decision process with 0/1 sparse rewards, which is $(\mathscr{S}, \mathscr{O}, \mathscr{A}, P(s'|s,a), \mathscr{G}^{(s)}, \mathscr{G}^{(o)}, \mathscr{T}, r(s,a,g), \gamma)$. $\mathscr{S}$ is the low-dimensional state space, $\mathscr{O}$ is the visual observation space, $\mathscr{A}$ is the action space, $\mathscr{G}^{(s)}$ and $\mathscr{G}^{(o)}$ are the goal spaces indicating the desired state and image. $P(s'|s,a)$ is the probability of the transition from state s to state s' after taking action a, and γ is the discounted factor. The reward function $r(s,a,g)$ is 1 only if the current state s reaches the goal g within some precision threshold and otherwise 0. $\mathscr{T}$ is the task space represented as a set of paired initial states s_0 and goals g from which to reset each episode. During the iterative data generation process of our framework, the task set varies among different rounds. An episode terminates either when

the goal is achieved or the agent reaches a maximum number of steps. We train a state-input agent $\pi(a|s, g^{(s)})$ for the data generation purpose during task expansion and finally distill to a visual policy $\pi(a|o, g^{(o)})$. We will use x to denote state s or image o input in the following.

We adopt a model-free actor-critic algorithm PPO [42] to train the goal-conditioned RL policy $\pi_\theta(a|x, g^{(x)})$ parameterized by θ and an universal value function [43] $V_\varphi(x, g^{(x)})$.

We also adopt behavior cloning (BC) to train the policy from a dataset $\mathscr{D}$ of goal-conditioned state-action or image-action pairs. The objective to minimize is

$$L_{\text{bc}} = \mathbb{E}_{(x,a,g^{(x)})\sim\mathscr{D}}[-\log \pi_\theta(a|x, g^{(x)})]. \tag{1}$$

IV. Method

The robot self-teaching paradigm starts from learning basic tasks from a seeding dataset that might be collected by humans, then progressively creates more data by the agent itself to facilitate generalizing to complex problems. To self-generate meaningful behaviors, we propose task expansion that could compose learned policies in state space to create data for more complex long-horizon tasks. A visual policy is then distilled from the self-generated massive data and could generalize to more complex tasks in a zero-shot manner. The overall framework is illustrated in Fig. 1. We introduce three parts of RST: warm-starting basic policy from a seeding dataset, self-generating data with task expansion, and distillation of a visual policy from the generated dataset.

A. Kickstart learning from a seeding dataset

RST starts from learning a seeding policy from an existing dataset. In the compositional manipulation setting we consider, there are exponentially many combinations of subtasks thus infeasible for a human labeler to demonstrate full trajectories. Instead, we only expect the dataset to contain basic behaviors interacting with single objects, such as pick-and-place, pulling a handle, and rotating a knob. In the block-stacking domain, the initial dataset comes from a scripted policy to transport cuboids; in the kitchen domain, we chunk the publicly available offline dataset into shorter trajectories demonstrating primitive skills. We train a goal-conditioned policy by BC over these trajectories, then robustify it with PPO [42] over the basic tasks in the seeding dataset.

B. Self-generating data via task expansion

In order to obtain a robot capable of more complex tasks than the ones provided in the seeding dataset without expensive expert labeling, we let the agent augment the dataset with self-generated trajectories. To generate meaningful novel tasks and trajectories, we propose task expansion that expands out more complex tasks from a successful trajectory by composing existing policies to it. The agent runs task expansion for multiple iterations to progressively generate data with higher levels of complexity, and restores all generated data for the final visual distillation stage.

Each iteration of task expansion works as follows. Given a successful trajectory with initial state s_0 and terminal state s_T, we aim to extend it to a more complex trajectory from s_0 to a new goal g by composing another learned strategy after s_T. As shown in Fig. 2, we adopt a sampling-based method to expand the goal. For each successful trajectory, we randomly sample a bunch of candidate goals from the goal space, then select the best new goal g^* based on a metric defined over the universal value function $V_\varphi(s, g)$ by

$$g^* = \arg\max_g V_\varphi(s_T, g) - V_\varphi(s_0, g). \tag{2}$$

Since the learned universal value function approximates the reachability between states under the 0/1 sparse reward setting, the metric would encourage selecting goals that are reachable from s_T while being non-trivial to solve from s_0. In the illustrated example, the original successful trajectory is moving the kettle forward. A new goal g' that moves the kettle back will not be selected since the agent already learns to move the kettle and the resulting task (s_0, g') cannot effectively expand the range of solvable task range of the agent. Another goal that additionally turns the bottom burner on is selected, which leads to a new task involving manipulating two components. The agent then executes the state-based policy to generate a trajectory from the original terminal state s_T to the discovered new goal $g^\star$. If the rollout is successful, it combines the trajectory with the original one and obtains a demonstration for a novel task $(s_0, g^\star)$.

The expanded trajectories are successful but might not be optimal for the newly discovered tasks $(s_0, g^\star)$, thus we further improve the data quality by training the state-based policy over these tasks. The policy is first trained with behavior cloning (BC) using expanded trajectories to get a good starting point. We again adopt PPO [42] to finetune this policy on generated tasks so as to revise minor errors after BC. We then execute the tuned policy on the tasks discovered in task expansion and collect all the states, visual observations, and actions as the self-generated dataset for future visual policy learning.

The newly generated tasks and their solutions would trigger the next round of task expansion and induce a more skillful policy to create data with growing complexity. With multiple rounds of task expansion, the agent will accumulate a massive amount of data demonstrating complex tasks.

Implementation: When sampling candidate goal states during task expansion, we incorporate minimal domain knowledge of object-centric environments, by only perturbing the states of one object in the original terminal state to create goal states.

C. Distill a generalizable visual policy

After self-generating a huge dataset containing complex behaviors from multiple rounds of task expansion, we can distill a generalizable visual-based policy using the image observations, image goals and corresponding actions predicted by the state policy stored in the dataset. The visual policy first processes the image observations and goals with an encoder specific to different domains and optionally concatenate them with other proprioceptive states to predict actions, and is trained with imitation learning. Other offline RL methods could also be applied but are not the focus of this work.

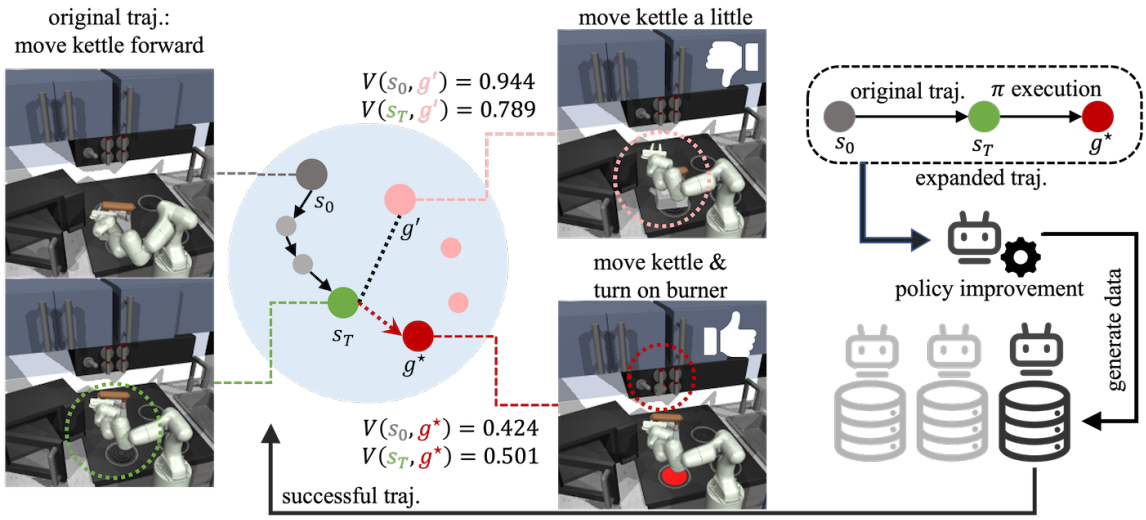


Fig. 2: Illustration of task expansion. Given a successful trajectory $s_0 \to s_T$, the agent first searches for a new task $g^\star$ that is reachable from s_T but hard to solve from s_0, then expands the trajectory by executing its policy from s_T to $g^\star$. The agent improves its policy with expanded trajectories and stores its own rollouts for the next round of task expansion.

The trained visual policy is directly applied to unseen evaluation tasks that require sequentially accomplishing multiple subtasks *without any fine-tuning* on these target tasks.

V. Experiment

We conduct empirical studies in two visual manipulation domains. We first experiment in a block-stacking domain to see whether RST can discover building structures by itself and give construction plans when tested over goal images of human-designed structures. We then apply RST to a standard offline RL benchmark "Franka kitchen", where RST achieves an impressive success rate on unseen multi-stage evaluation tasks. The experiments are repeated for 3 seeds.

A. Block-stacking domain

In the multi-object stacking domain with a maximum of 6 cuboid blocks on a desk, the robot aims to move objects to their corresponding target positions and meanwhile keep them stable there. In data-generation policy training, the goal is specified as target poses of some blocks. The target positions can be in the air, which requires some non-target blocks stacked below. Each action selects one object and its desired pose and directly teleports that object in the simulator, similar to the setting in [44]. The agent only receives a non-zero reward when all targets are reached stably. In task expansion, the agent samples new tasks by adding one target object, modifying one target to a new position, or associating one target with another object.

When distilling a visual policy for validation in simulation, we adopt a slot-attention [45] module to encode images as object-centric features, then fuse them with a Transformer [46] to predict actions. The slot-attention encoder is trained over all images in the generated dataset with an unsupervised reconstruction loss and is then fixed when training other parts of the visual policy with behavior cloning. For deployment to the real world, we add intensive domain randomization to the visual properties of the simulation, and instead train a diffusion policy [47] with an ResNet18-based encoder [48] from scratch in distillation.

Data source	"I" sr. (%)	"3T" sr. (%)	"Y" sr. (%)
round 1	0.7±1.2	4.0±1.7	0.0±0.0
round 2	0.7±0.6	2.0±1.0	0.0±0.0
round 1, 2	9.7±1.5	19.0±4.4	1.6±0.6
round 3	10.0±3.6	25.3±5.8	1.3±0.6
round 1, 2, 3	**28.0±1.7**	**41.7±4.0**	**15.0±3.0**
Direct RL	0.0±0.0	0.0±0.0	0.0±0.0

TABLE I: Zero-shot success rate and the standard deviation of our visual stacking policy distilled from datasets generated in task expansion and the result of directly training RL on these evaluation tasks. The policies are evaluated over three categories of unseen structures. "Direct RL" fails due to the hard-exploration issue in our binary sparse reward setting.

a) Main results: After collecting data with three rounds of task expansion, we distill a visual policy and test whether it can generalize to challenging compositional tasks that are never provided to it before. As shown in Table I, we design three categories of goals named "I", "3T" and "Y", and test from initial states with all blocks randomly scattered on the desk. We try distilling visual policies from different portions of the self-generated data and test each policy by rolling out 100 episodes for every evaluation goal. The zero-shot performance on all the evaluation tasks gets better using data generated from later rounds of task expansion. The best result is achieved using the full data from all three rounds.

We also compare with a naive RL method which first

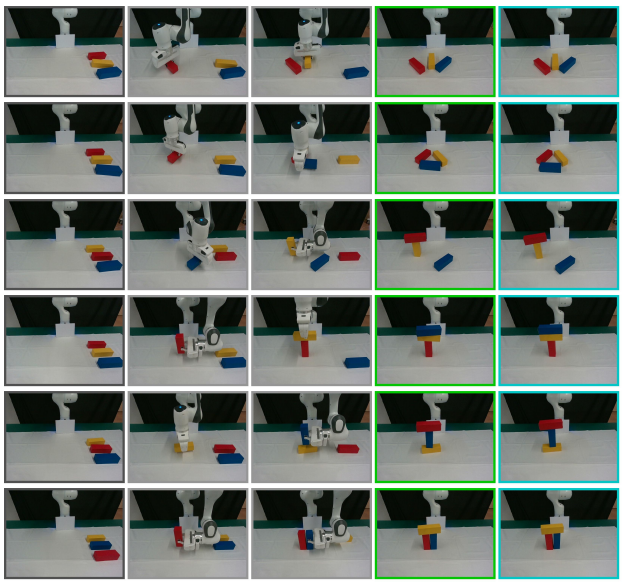


Fig. 3: Strategies in image-based block-stacking evaluation tasks. Each row shows how our agent solves one type of task. The first four columns show the observations along rollout trajectories and the last column is the goal image.

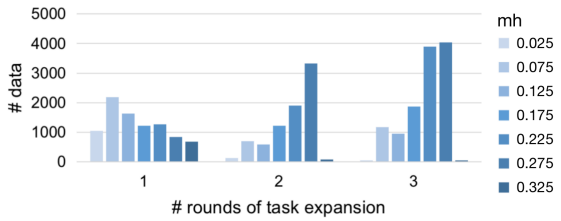


Fig. 4: Distribution of maximum heights (mh) of target objects in generated tasks during task expansion in the block-stacking domain. The generated tasks gradually shift to higher structures with more expansion rounds, indicating more blocks stacked together.

Methods	"I" sr. (%)	"3T" sr. (%)	"Y" sr. (%)
RST w/o restr.	6.6±3.8	0.7±0.4	0.2±0.1
RST init2new	14.5±8.1	0.1±0.1	0.4±0.3
RST end2new	13.4±9.7	2.5±4.2	1.0±0.9
RST	**70.8±18.6**	**44.3±6.7**	**8.0±7.0**

TABLE II: Success rate of zero-shot evaluation compared with other design choices. All variants use the policies after the first round of task expansion for evaluation.

learns to move single objects with BC from the same initial dataset as RST, then directly optimizes the state-based policy over the *evaluation* goal distribution with PPO. Since the agent can only get a positive reward when all the goals are reached which is extremely difficult to explore, it does not make any progress after 137M timesteps of training. Therefore, RST can be viewed as an automatic curriculum that smooths out the challenges of directly optimizing strong compositional problems under sparse reward.

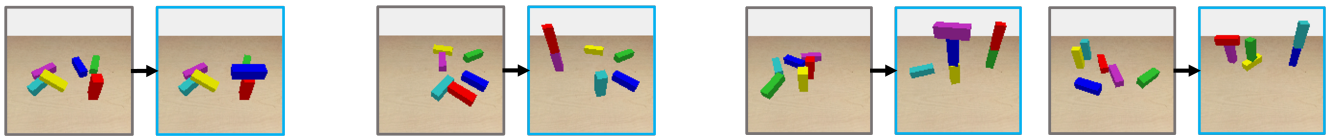
b) Deployment to the real world: To mitigate the sim-to-real gap, we randomize the textures, lighting and camera poses of the simulator and render the generated data as images to distill a vision policy. The policy is deployed to a real Franka panda robot to build various structures that match goal images. At each step, the policy takes as input the current image observation captured by a front-view camera, as well as the goal image depicting the desired pose for all blocks. By incorporating the goal image as input, the visual policy is tailored for direct generalization to new structures. The learned strategy is illustrated in Fig. 3, where the policy execution is in the first four columns and the goal images are in the last column. The visual policy is capable of both easier goal configurations that simply rearrange blocks on the table surface (the first two rows), and harder structures with the blocks stacked into various shapes (the last four rows).

c) Analysis of generated data in task expansion: Figure 4 shows the distribution of the maximum height of the target positions in each task for each round of task expansion respectively, which presents a tendency towards higher targets as the number of rounds increases. We also illustrate samples from different rounds of task expansion in the self-generated dataset. RST generates progressively more complex data as shown in Fig. 5. In the first round of expansion, the agent discovers a single "T" shape. It becomes capable of building multiple towers and stacking more cuboids together in later rounds.

d) Ablation studies: We ablate the restriction of only allowing one goal to be different from the original task by sampling new goals randomly in the goal space ("w/o restr."). As shown in Table II, it performs worse than our presented method, possibly because goal sampling with the restriction could generate more feasible tasks. We study other metrics to select the best new goal in task expansion by choosing $\arg\max_g V_\varphi(s_T, g)$ ("end2new") or $\arg\min_g V_\varphi(s_0, g)$ ("init2new"). As is shown in Table II, the variants with "init2new" and "end2new" metrics both generalize significantly worse. These results imply that generating new tasks that are both feasible by composing previous policies and challenging enough for the current agent to improve is critical to the success of RST.

B. Franka kitchen domain

In the "Franka kitchen" environment from D4RL [49], The agent controls a Franka robot to manipulate components in a simulated kitchen, including a microwave ("M"), kettle ("K"), light switch ("L"), two burners ("B" and "T"), and two cabinets ("S" and "H"). The states and goals are defined over the degrees of freedom of the robot and objects. The visual observations and goals are third-person-view images. The action space is the 9 joints of the robot. In the original benchmark, the initial states and goals are sampled from a predefined distribution. During self-data generation, we let the environment reset from starting and desired states proposed by our agent. A trajectory is considered as successful and the agent gets a reward of 1 only if all the elements reach their desired states, otherwise, the reward signal is 0.



(a) The first round (b) The second round (c) The third round

Fig. 5: Generated data (initial state →goal state) during different rounds of task expansion. Progressively more challenging structures that require moving more cuboids are generated by our agent.

	HM	MT	HLM	BKT	BLM	LMST	BLST	HLMT
RST	0.99±0.01	0.93±0.03	0.51±0.02	0.45±0.10	0.52±0.04	0.72±0.05	0.46±0.22	0.52±0.15
FLAP [22]	0.00±0.00	0.09±0.01	0.00±0.00	0.00±0.00	0.01±0.01	0.00±0.00	0.00±0.00	0.00±0.00
State @ 1^{st} rd.	0.31±0.43	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
State @ 4^{th} rd.	0.45±0.33	0.99±0.01	0.24±0.26	0.00±0.00	0.07±0.10	0.33±0.20	0.47±0.30	0.41±0.22
State @ 8^{th} rd.	0.97±0.02	0.99±0.00	0.43±0.23	0.00±0.00	0.91±0.06	0.70±0.15	0.50±0.24	0.80±0.09

TABLE III: Mean and std. of zero-shot success rate on test tasks composed of different combinations of components to interact with in Franka Kitchen. Each number is evaluated with 100 trials. "RST" denotes the distilled visual policy.

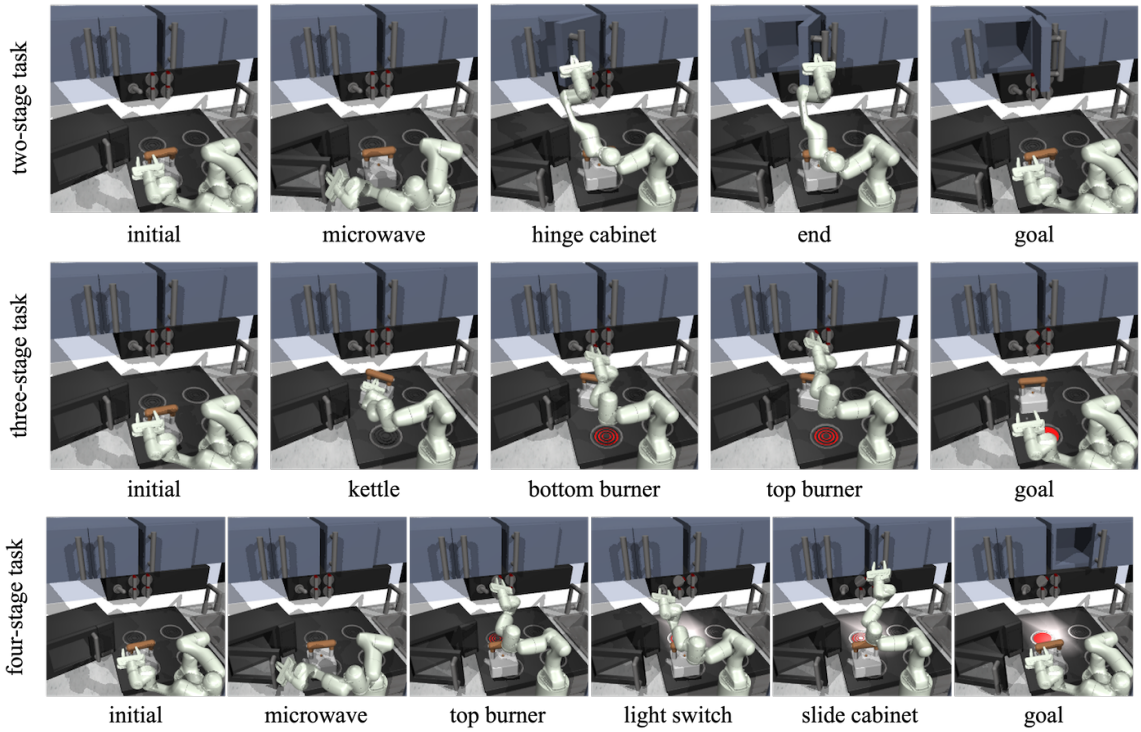


Fig. 6: Rollouts of the learned visual policy for solving evaluation tasks that require sequentially manipulating two, three and four components in the Franka kitchen domain. The goal images are shown in the rightmost column.

We split the "kitchen-partial" dataset into shorter chunks containing interaction with single components and relabel each chunk as a successful demo of the task $(s_i, g = s_j)$ to form the seeding dataset. During task expansion, We randomly perturb the achieved state of one component in the kitchen as sampled new goals. We run 8 rounds of expansion in total to generate a large dataset. In visual distillation, we adopt pre-trained R3M [50] to process the images.

a) Main results: To validate the compositional generalization ability, we specify several evaluation tasks that require sequentially interacting with multiple components, and are named with abbreviations of components that should be moved. In Table III, we report the zero-shot evaluation success rate of an image-based agent distilled from the last three rounds of generated data, and also the performance of the state-based agent after 1, 4, and 8 round(s) of expansion as reference. Our image-based agent (first row) achieves impressive performance over a broad range of compositional tasks that require manipulating 2, 3, or 4 components. The success rates of our state-based agent after different rounds of expansion (last three rows) show that the agent gradually generalizes to problems with increasing difficulty. Interestingly, our image-based agent sometimes demonstrates better zero-shot generalization ability than the state-based ones (e.g., the "BKT" task) possibly due to the benefit of multi-task training over the generated broad dataset.

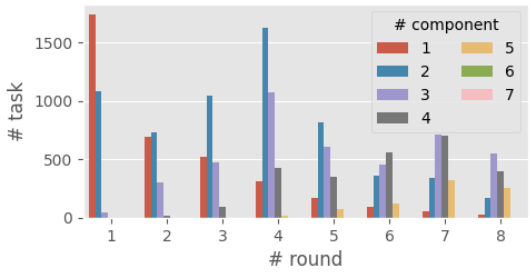


Fig. 7: Statistics of generated data in Franka kitchen. Each bar shows the number of generated tasks that require manipulating a specific number of components. RST discovers more complex tasks with more rounds of expansion.

	Initial	Till 1^{st} rd.	Till 4^{th} rd.	Till 8^{th} rd.
# data	133k	323k	1.059M	1.799M

TABLE IV: The accumulated number of self-generated state-action pairs in different rounds of expansion in RST.

We compare with FLAP [22], an offline method that learns goal-conditioned skills and an affordance model from datasets and then plans subtasks to solve multi-stage problems. Following their original setting of learning from a dataset containing primitive robot skills, we use our seeding dataset to train FLAP. We adopt the same R3M [50] encoder for FLAP as our method. With our best efforts of tuning, FLAP gets some zero-shot success on two-stage tasks but fails completely on tasks with four stages. We also try to tune FLAP by allowing it to do online interactions over the *evaluation* tasks, but it fails to make progress, possibly because its planning and affordance model learns poorly from a small seeding dataset. We argue that the performance of such planning-based methods could be largely limited by the capability of a planner to break down complex problems, while RST self-generates data with growing complexity and can scale up to compositional problems better.

b) More analysis: The number of components that differ in initial and goal states in self-generated data is shown in Fig. 7. In the early rounds of task expansion, most discovered tasks only perturb one or two components. After several rounds, complex tasks that require interacting with four or more components start to emerge. We also report the accumulated number of data generated by RST after different rounds of expansion in Table IV. Initially, there are only 133k steps in the seeding dataset. RST progressively creates more data in each round, inducing a rich dataset with $> 10\times$ more data than the initial one provided to it.

VI. Conclusion

We present robot self-teaching, a novel framework that self-generates effective tasks and trajectories to train a generalizable visual control policy in multi-object manipulation scenarios, instead of relying on datasets purely collected by humans. At the core of RST is task expansion, which progressively creates data with increasing complexity by composing previously learned tasks. RST focuses on compositional generalization currently, and complementary data augmentation techniques [51] could be integrated to enable other aspects of generalization. We believe leveraging self-generated data is a promising direction for pre-training general robot control policies to tackle real-world problems.

References

[1] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[2] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *arXiv preprint arXiv:2205.11487*, 2022.

[3] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[5] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[6] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[7] S. Lange, T. Gabel, and M. Riedmiller, "Batch reinforcement learning," *Reinforcement learning: State-of-the-art*, pp. 45–73, 2012.

[8] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International conference on machine learning*. PMLR, 2019, pp. 2052–2062.

[9] A. Kumar, A. Singh, F. Ebert, Y. Yang, C. Finn, and S. Levine, "Pre-training for robots: Offline rl enables learning new tasks from a handful of trials," *arXiv preprint arXiv:2210.05178*, 2022.

[10] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[11] Y. Chebotar, K. Hausman, Y. Lu, T. Xiao, D. Kalashnikov, J. Varley, A. Irpan, B. Eysenbach, R. C. Julian, C. Finn, *et al.*, "Actionable models: Unsupervised offline reinforcement learning of robotic skills," in *International Conference on Machine Learning*. PMLR, 2021, pp. 1518–1528.

[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*. PMLR, 2022, pp. 991–1002.

[13] 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.

[14] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," in *Conference on robot learning*. PMLR, 2020, pp. 1113–1132.

[15] 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.

[16] Y. Dai, M. Yang, B. Dai, H. Dai, O. Nachum, J. Tenenbaum, D. Schuurmans, and P. Abbeel, "Learning universal policies via text-guided video generation," *arXiv preprint arXiv:2302.00111*, 2023.

[17] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[18] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," *Advances in neural information processing systems*, vol. 34, pp. 20 132–20 145, 2021.

[19] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in neural information processing systems*, vol. 34, pp. 15 084–15 097, 2021.

[20] S. Lee, Y. Seo, K. Lee, P. Abbeel, and J. Shin, "Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble," in *Conference on Robot Learning*. PMLR, 2022, pp. 1702–1712.

[21] K. Fang, P. Yin, A. Nair, and S. Levine, "Planning to practice: Efficient online fine-tuning by composing goals in latent space," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 4076–4083.

[22] K. Fang, P. Yin, A. Nair, H. R. Walke, G. Yan, and S. Levine, "Generalization with lossy affordances: Leveraging broad offline data for learning visuomotor tasks," in *6th Annual Conference on Robot Learning*, 2022.

[23] S. Park, D. Ghosh, B. Eysenbach, and S. Levine, "Hiql: Offline goal-conditioned rl with latent states as actions," *arXiv preprint arXiv:2307.11949*, 2023.

[24] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," in *6th Annual Conference on Robot Learning*, 2022.

[25] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, *et al.*, "Inner monologue: Embodied reasoning through planning with language models," in *Conference on Robot Learning*. PMLR, 2023, pp. 1769–1782.

[26] A. Zeng, A. Wong, S. Welker, K. Choromanski, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, *et al.*, "Socratic models: Composing zero-shot multimodal reasoning with language," *arXiv preprint arXiv:2204.00598*, 2022.

[27] J. Zhang, J. Zhang, K. Pertsch, Z. Liu, X. Ren, M. Chang, S.-H. Sun, and J. J. Lim, "Bootstrap your own skills: Learning to solve new tasks with large language model guidance," in *Conference on Robot Learning*. PMLR, 2023, pp. 302–325.

[28] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[29] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, "Reverse curriculum generation for reinforcement learning," in *Conference on robot learning*. PMLR, 2017, pp. 482–495.

[30] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," in *International conference on machine learning*. PMLR, 2018, pp. 1515–1528.

[31] Z. Ren, K. Dong, Y. Zhou, Q. Liu, and J. Peng, "Exploration via hindsight goal generation," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[32] S. Racaniere, A. Lampinen, A. Santoro, D. Reichert, V. Firoiu, and T. Lillicrap, "Automated curriculum generation through setter-solver interactions," in *International conference on learning representations*, 2019.

[33] R. Wang, J. Lehman, J. Clune, and K. O. Stanley, "Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions," *arXiv preprint arXiv:1901.01753*, 2019.

[34] R. Wang, J. Lehman, A. Rawal, J. Zhi, Y. Li, J. Clune, and K. Stanley, "Enhanced poet: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9940–9951.

[35] M. Dennis, N. Jaques, E. Vinitsky, A. Bayen, S. Russell, A. Critch, and S. Levine, "Emergent complexity and zero-shot transfer via unsupervised environment design," *Advances in neural information processing systems*, vol. 33, pp. 13 049–13 061, 2020.

[36] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging procedural generation to benchmark reinforcement learning," in *International conference on machine learning*. PMLR, 2020, pp. 2048–2056.

[37] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Conference on Robot Learning*. PMLR, 2020, pp. 66–75.

[38] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[39] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021.

[40] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[41] T. Chen, J. Xu, and P. Agrawal, "A system for general in-hand object re-orientation," *Conference on Robot Learning*, 2021.

[42] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[43] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *International conference on machine learning*. PMLR, 2015, pp. 1312–1320.

[44] Y. Li, T. Kong, L. Li, Y. Li, and Y. Wu, "Learning to design and construct bridge without blueprint," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 2398–2405.

[45] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf, "Object-centric learning with slot attention," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 525–11 538, 2020.

[46] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[47] 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.

[48] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[49] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[50] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3m: A universal visual representation for robot manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 892–909.

[51] T. Yu, T. Xiao, A. Stone, J. Tompson, A. Brohan, S. Wang, J. Singh, C. Tan, J. Peralta, B. Ichter, *et al.*, "Scaling robot learning with semantically imagined experience," *arXiv preprint arXiv:2302.11550*, 2023.