An Object-Oriented Approach for Generating Low-Fidelity Environments for Curriculum Schema Transfer

Yash Shukla[1], Kaleb Loar[2], Robert Wright[3], and Jivko Sinapov[1]

***Abstract*— Advances in reinforcement learning (RL) have enabled robots to learn a wide range of behaviors. Despite this, scaling the magnitude of learned behaviors to complex sequential decision making tasks is still computationally expensive because of sample inefficiency of many RL approaches. A viable solution is to learn the complex task through a curriculum, by optimizing the sequence of interactions of the robot. Generating the curriculum is still non-trivial and requires extensive experimentation. In this work, we provide an object-oriented approach to characterize an environment, and then provide a framework to generate a low-fidelity (LF) environment for a complex robotic high-fidelity (HF) environment. We show that an object-oriented approach helps to conveniently transfer the schema of the optimized curriculum from the LF environment to the complex HF environment. We demonstrate that our approach yields sample efficient learning in robotic navigation and robotic manipulation domains, where the robot has to perform a series of sequential decisions to achieve the target.**

I. Introduction

Deep reinforcement learning utilizes the learning capacity of neural networks to learn complex tasks ranging from Atari games to robot manipulation tasks [1], [2]. Most existing methods systems suffer from catastrophic forgetting, which is the tendency to forget previously learned behavior while learning a new behavior [3]. This is a serious problem in sequential decision making tasks, in which an agent needs to perform a series of sequential behaviors to reach a desired goal, fairly common in robotic settings. Curriculum Learning (CL) is a tool that tries to mitigate this problem by gradually increasing the complexity of tasks the agent tries to learn, giving the agent sufficient interaction experience to refocus on the initially learned behaviors [4]. The core of CL is to generalize the experience and knowledge acquired in simple tasks and leverage it to learn complex tasks, thereby increasing performance and reducing training time [5]. A major limitation of many CL approaches is that the time to generate the curriculum is greater than the time to learn the target task from scratch, which prohibits the use of such methods in complex, real-world, high-fidelity domains [5]. This makes it infeasible to optimize the curriculum in a high-fidelity robotic domain, given the costly setup. Sim2Real Transfer [6], [7] allows a model trained in a simulation to be deployed on a physical robot. However, it suffers from "Reality Gap" [6], where the simulation policy performs poorly on transfer. Continual learning on incrementally realistic simulations may help to mitigate this problem, as the agent attempts to abstract and transfer relevant information [9]. One key limitation of Sim2Real approaches is the need for a simulation whose task Markov Decision Process (MDP) representation exactly matches the complex dynamics of the realistic scenario, and is not always feasible [10].

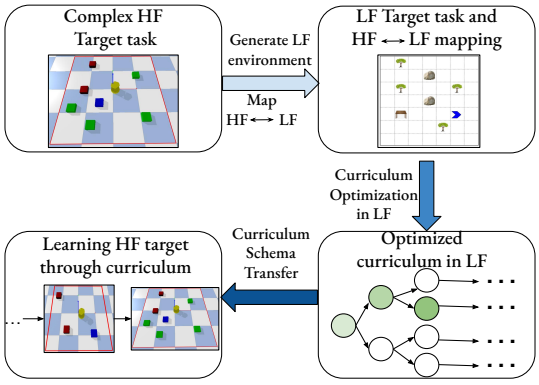


Fig. 1: Overview of the curriculum schema transfer procedure. This work focuses on generating the low-fidelity environment given a high-fidelity environment and attaining the mapping function

A potential solution for generating a curriculum in a high-fidelity (HF) environment is to formulate an optimized curriculum of the task in a simplified version of the HF environment, i.e., a low-fidelity (LF) environment [11] (Fig 1). Then, for each task in the curriculum generated in the LF environment, the task parameters are transferred to generate a corresponding curriculum in the high-fidelity environment. This is termed as the *curriculum schema transfer* approach. Optimizing the curriculum in the LF environment provides a convenient, faster and sample efficient way of generating the curriculum and learning the goal task [11]. This handles cases in which Sim2Real fails i.e. when the simulation model has different MDP representation than the physical model.

One key limitation of this approach is the lack of a set of guidelines that describe in detail what a suitable LF environment is, and how this LF environment maps to the HF environment. In this work, we tackle the low-fidelity environment generation in the curriculum schema transfer problem, by proposing an object-oriented MDP based approach [17], [18] to obtain the LF environment and the LF to HF mapping function for a given HF domain. We perform extensive evaluation on a sequential robotic navigation domain and a robotic manipulation domain. The navigation domain is inspired by *Minecraft*, in which a TurtleBot aims to craft a Stone-Axe by navigating in an arena and collecting pieces of trees and rocks scattered in the environment, where as the manipulation domain involves a

[1]Department of Computer Science, Tufts University, {yash.shukla, jivko.sinapov}@tufts.edu
[2]Assured Information Security, Inc., loark@ainfosec.com
[3] Georgia Tech Research Institute, robert.wright@gtri.gatech.edu

Panda arm picking items and placing them at target locations. Through experiments, we show that optimizing curriculum in the generated LF environment saves learning time and helps scale the magnitude of behaviors learned by an RL agent in a robotic setting by proposing an optimized curriculum, compared to learning without any curriculum.

II. Theoretical Framework

A. Markov Decision Processes

An episodic Markov Decision Process (MDP) M is defined as a tuple $(\mathcal{S},\mathcal{A},p,r,\gamma,\mathcal{S}_0,\mathcal{S}_f)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $p(s'|s,a)$ is the transition function, $r(s',a,s)$ is the reward function and $\gamma \in [0,1]$ is the discount factor. At each timestep t, the agent observes a state s and performs an action a given by its policy function $\pi_\theta(a|s)$, with parameters θ. The agent's goal is to learn an *optimal policy* π^*, maximizing its discounted return $G_0 = \sum_{k=0}^{K}\gamma^k r(s_{k+1},a_k,s_k)$ until the end of the episode at timestep K. $\mathcal{S}_0$ and $\mathcal{S}_f$ are the sets of starting states and terminal states, respectively.

B. Curriculum Learning (CL)

We define a task-level curriculum as:
Let $\mathcal{T}$ be a set of tasks, where $M_i = (\mathcal{S}_i, \mathcal{A}_i, p_i, r_i, \mathcal{S}_{0_i}, \mathcal{S}_{f_i})$ is a task in $\mathcal{T}$. Let $\mathcal{D}^{\mathcal{T}}$ be the set of all possible transition samples from tasks in $\mathcal{T}$: $\mathcal{D}^{\mathcal{T}} = \{(s,a,r,s') \,|\, \exists\, M_i \in \mathcal{T}\ s.t.\ s \in \mathcal{S}_i,\ a \in \mathcal{A}_i,\ s' \sim p_i(\cdot|s,a),\ r \leftarrow r_i(s,a,s')\}$. A curriculum $C = [M_1, M_2, \ldots, M_U]$ is an ordered list of tasks, where M_i is the i^{th} task in the curriculum. The ordered list signifies that samples from M_i must be used for training a policy before samples from M_{i+1} are used. The sequence of tasks terminates on the target task M_U.

C. Object-Oriented MDP

The Object-Oriented MDP (OOMDP) representation [17] abstracts the task description, enabling us to intuitively generalize the HF environment. In OOMDP setting, the task space is abstracted by a set of classes $\mathcal{C} = \{C_1, C_2, \ldots\}$, and each class $C_i \in \mathcal{C}$ has a set of parameters $Par(C_i) = \{C_i.p_1, \ldots, C_i.p_{|p|}\}$. Each parameter $C_i.p_i$ has a range of values the parameter can attain, given by $Range(C_i.p_i)$. At any given instance, an environment consists of a set of objects $\mathcal{O} = \{o_1, o_2, \ldots, o_{|o|}\}$, where each object o_i is an instance of a class C_i and is defined by the values of the parameters for that class $o_i = Par(C_i(o_i))$. For a task, the OOMDP state s_{oo} is given by the union of all object states $s_{oo} = \bigcup_{o_i \in \mathcal{O}} State(o_i)$, where each object state is the value of the parameters of that object $State(o_i) = \{o_i.p_1, \ldots, o_i.p_{|p|}\}$. In our setting, a one-to-many mapping exists between the OOMDP state and the MDP state.

D. Problem Formulation

CL aims to generate a curriculum and train an agent on a sequence of tasks $\{M_1, M_2, ..., M_U\}$, such that the agent's convergence performance increases, or its time-to-threshold performance on the final target task (M_U) improves relative to learning from scratch. Time-to-threshold (Δ) metric computes how much faster an agent can learn a policy that achieves expected return $G \geq \delta$ on the target task if it transfers knowledge, as opposed to learning from another approach. Here δ is desired performance threshold, in success rate or episodic reward. The domain $\mathcal{T}^{HF}$ of possible tasks is a set of MDPs in the high-fidelity (HF) environment, obtained by varying the objects in the target task $\mathcal{O}_{M_i^{HF}}$, their parameters and the goal condition $g_{M_i^{HF}}$ for task M_i^{HF}.

A suitable candidate for a Low-Fidelity environment is one that has a corresponding OOMDP state s_{oo}^{LF} for any HF OOMDP state s_{oo}^{HF}. To be qualified as a Low-Fidelity environment, the time-to-threshold for the LF task M_i^{LF} must be lower than the time-to-threshold for the equivalent HF task M_i^{HF}, i.e. $\Delta(M_i^{LF} - M_i^{HF}) > 0$.

The goal of this paper is two-fold:

1) Assuming existence of an LF environment, find a LF environment that will satisy the above constraints.
2) Assuming existence of a set of mapping functions $\mathcal{F} := \{f_1, \ldots, f_n\}$ that provide a convenient mapping between the OOMDP states of the LF and the HF environments, the goal is to find one such mapping.

Thus, after obtaining an appropriate LF environment for the final target task of the HF environment, a curriculum can be generated and optimized in this LF environment, with inexpensive setup and easier experimentation [11].

Let $\mathcal{C}_U^{\mathcal{T}_{LF}}$ be the set of all curricula over tasks $\mathcal{T}^{LF}$ of length U in the LF environment. Similarly, let $\mathcal{C}_U^{\mathcal{T}_{HF}}$ be the set of all curricula over tasks $\mathcal{T}^{HF}$ of length U in the HF environment. Each task of the optimized curriculum in LF ${c_U}^{\mathcal{T}_{LF}}$ is mapped through $\mathcal{F}$ to obtain an optimized curriculum for HF ${c_U}^{\mathcal{T}_{HF}}$, where the mapping is an affine transformation given by:

$$s_{oo}^{HF} = K \odot s_{oo}^{LF} + L$$

where $K = [k_1, \ldots, k_n]^T \in \mathbb{R}^n$ and $L = [l_1, \ldots, l_n]^T \in \mathbb{R}^n$ denote linear mapping and translation vectors and $\odot$ is the Hadamard product. Thus, a parameter mapping $(f_i : o_i.a_i^{LF} \rightarrow o_i.a_i^{HF})$ is given by:

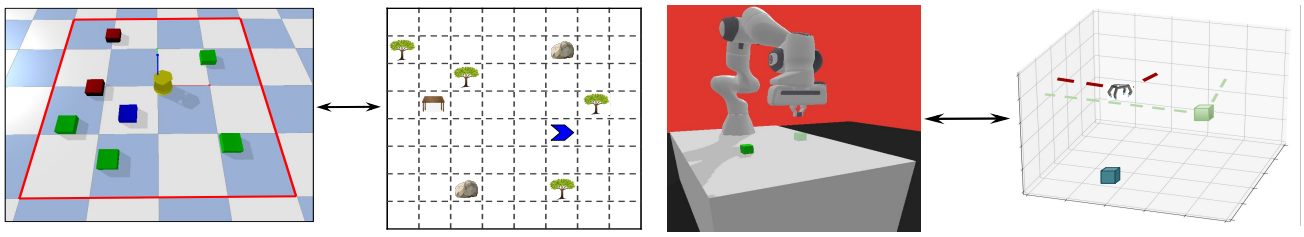
$$o_i.p_i^{HF} = k_i.(o_i.p_i^{LF}) + l_i$$

We assume that the source tasks of the curriculum in the HF environment are learned before learning the final target task, as described in Section II-B.

III. Low-Fidelity Environment Generation

The curriculum generation for the high-fidelity (HF) task consists of three parts: (1) Generation of a suitable low-fidelity (LF) environment along with its HF mapping functions $(\mathcal{F})$; (2) curriculum optimization in the LF environment, and (3) curriculum schema transfer to the HF environment. We provide a framework for obtaining a suitable LF environment, and generating the mapping functions according to constraints described in Section II-D.

To generate a suitable LF environment, we begin by modelling the HF target task using OOMDP approach [17], [18]. The HF environment is modelled using a set of classes $\mathcal{C}^{HF} = \{C_1^{HF}, \ldots, C_{|c|}^{HF}\}$, and each class



(a) HF (left) and LF (right) target task for Crafter-TurtleBot domain

(b) HF (left) and LF (right) target task in the Panda-Pick-and-Place domain

Fig. 2: Illustration of the final target task in the high-fideity (HF) and low-fidelity (LF) environments for a Crafter-TurtleBot domain and Panda-Pick-and-Place manipulation domain. The goal of the agent in the navigation domain is to craft a pogo stick at the crafting table by collecting two trees and a rock, whereas in the manipulation domain, the goal is to pick an object and place it at a target location.

Algorithm 1 `LF_Gen`$(\mathcal{C}^{HF}, \mathcal{P}^{HF}, \mathcal{O}_{M_U^{HF}}, \mathcal{X}, n)$

Output: LF target task descriptors: $\mathcal{C}^{LF}, \mathcal{P}^{LF}, \mathcal{O}_{M_U^{HF}}$
Set of mapping functions: $\mathcal{F}$
Placeholder Initialization: Set of Classes in LF: $\mathcal{C}^{LF} \leftarrow \emptyset$
Set of Parameters for each class in LF: $\mathcal{P}^{LF} \leftarrow \emptyset$
Set of Objects in LF target task: $\mathcal{O}_U^{LF} \leftarrow \emptyset$
Set of Mapping functions $\mathcal{F} \leftarrow \emptyset$
Algorithm:

1: $\mathcal{C}^{LF} = \{c \mid c \in \mathcal{C}^{HF}\}$ ▷ Assign set of classes in LF
2: **for** $o \in \mathcal{O}_{M_U^{HF}}$ **do**
3: $c \leftarrow$ `getBaseClass`(o)
4: $\mathcal{O}_{M_U^{LF}} \leftarrow \mathcal{O}_{M_U^{LF}} \cup \{$`createObject(c)`$\}$ ▷ Create objects in LF
5: $\mathcal{P}_c^{HF} = \mathcal{P}^{HF}[c]$
6: **for** $p \in \mathcal{P}_c^{HF}$ **do**
7: $\mathcal{P}_c^{LF} \leftarrow \mathcal{P}_c^{LF} \cup \{p\}$; $\mathcal{P}^{LF} \leftarrow \mathcal{P}^{LF} \cup \mathcal{P}_c^{LF}$
8: **if** $|Range(o.p)| \leq n$ **then** ▷ Compare parameter range with maximum permissible cardinality constant given by human expert
9: $o.p^{LF} = o.p$
10: $f_{o.p} = 1$
11: **else**
12: $o.p^{LF} = (o.p/\mathcal{X}[c,p])$ ▷ LF value given by hyperparam
13: $f_{o.p} = 1/\mathcal{X}[c,p]$
14: $\mathcal{O}_{M_U^{LF}}[o].p = o.p^{LF}$ ▷ Assign value to LF object
15: $\mathcal{F} \leftarrow \mathcal{F} \cup \{f_{o.p}\}$
16: **return** $\mathcal{C}^{LF}, \mathcal{P}^{LF}, \mathcal{O}_{M_U^{LF}}, \mathcal{F}$

$C_i^{HF} \in \mathcal{C}^{HF}$ has a set of parameters $Par(C_i^{HF}) = \{C_i^{HF}.p_1, C_i^{HF}.p_2, \ldots, C_i^{HF}.p_{|p|}\}$, each parameter $C_i^{HF}.p_i$ contained within a range $Range(C_i^{HF}.p_i)$. The target task in the HF environment, with MDP M_U^{HF}, has a set of objects $\mathcal{O}_{M_U^{HF}} = \{o_1^{HF}, o_2^{HF}, \ldots, o_{|o|}^{HF}\}$ with object states given by the value of the parameters of that object $State(o_i^{HF}) = \{o_i^{HF}.p_i, o_i^{HF}.p_2, \ldots, o_i^{HF}.p_{|p|}\}$.

The overall approach of generating the LF environment and the mapping function is given in Algorithm 1. The input to `LF_Gen` is the set of HF environment classes $\mathcal{C}^{HF}$, the set of set of parameters $\mathcal{P}^{HF}$, objects in the the HF target task $\mathcal{O}_{M_U^{HF}}$, a set of discretization hyperparameter constants $\mathcal{X}$, and a human-expert specified finite scalar n, denoting the maximum desired cardinality for the parameter range for the task in hand. The output of `LF_Gen` is the set of LF environment classes $\mathcal{C}^{LF}$, the set of set of parameters $\mathcal{P}^{LF}$, objects in the the LF target task $\mathcal{O}_{M_U^{LF}}$ (together making up the LF target task), along with the LF $\leftrightarrow$ HF mapping function set. To begin, all the classes in the HF environment have a corresponding class in the LF environment (line 1). For each object in the HF target task, `LF_Gen` first determines the base class for the object, and creates an LF object for the class (lines 2-4). Then, for each parameter of that class, `LF_Gen` determines if the range for the parameter lies with the desired cardinality for the task. If it does, then the value of the parameter of the HF target task object is equivalent with the value of the parameter of the LF target task object, and the mapping between these two parameters is 1 (lines 8-10). If the range of the parameter exceeds the desired cardinality, then the LF object parameter is determined by the discretization hyperparameter for the HF parameter $(\mathcal{X}[c,p])$, and the mapping is given by the reciprocal of the hyperparameter (lines 11-13).

Thus, by iterating over all parameters for all classes in the HF environment, `LF_Gen` bounds the range of the parameter space, thereby decreasing the number of OOMDP states, while keeping the objective of the task unchanged. This accelerates the curriculum optimization procedure, and provides a convenient mapping for curriculum schema transfer.

Once a suitable LF environment is generated, the next step entails optimizing the curriculum in LF environment. The curriculum in the LF is obtained using "ACuTE" [11], where the agent generates source tasks by varying the parameters of the objects present in the final target task, and uses beam search to optimize the source task sequence based on the transfer potential to the target task. Then, each source task of the curriculum in LF is mapped to a source task in HF using the set of mapping functions $(\mathcal{F})$.

IV. Experiments and Results

We test our approach on two complex robotic domains. **The Crafter-TurtleBot domain** (Fig 2a) involves a TurtleBot whose goal is to collect 2 trees and a rock and then craft a stone-axe at the crafting table. The agent needs to navigate, face the object and perform the break action to collect it in inventory. The set of classes in the HF environment are $\mathcal{C}_{nav}^{HF} = \{wall, trees, rocks, crafting\_tables\}$, and the class parameters are: $Par(wall) = \{height, width\}$, where these parameters can assume continuous values within $[1m, 3m]$; and $Par(trees) = \{no\_of\_trees\}$; $Par(rocks) = \{no\_of\_rocks\}$; $Par(crafting\_tables) = \{no\_of\_CT\}$, with these parameters taking values from a discrete range.

The agent is a TurtleBot, rendered in PyBullet [19]. In the HF environment, the agent's navigation actions are moving forward 0.25 units and rotating by $\frac{\pi}{9}$ radians. The

agent's sensor emits a beam at incremental angles of $\frac{\pi}{10}$ to determine the closest object in the angle of the beam. Two additional sensors provide information on the amount of wood and stones in the agent's inventory. The objects for the classes $\{trees, rocks, crafting\_tables\}$ assume continuous locations, chosen randomly, to more closely model the real-world. So an OOMDP state for an object will correspond to multiple MDP states, denoting a one-to-many mapping.

The set of classes in the LF environment is equivalent to the set of classes in the HF environment $\mathcal{C}^{LF}_{nav} = \{wall, trees, rocks, crafting\_tables\}$, but the range of values for all continuous parameters $(wall(height, width))$ are bounded within a discrete set $\{4, 5, ..., 12\}$, reducing the number of OOMDP states attainable. Thus, even though the navigable area in the HF environment can attain values within the continuous range $[1\times1{,}3\times3]m^2$, the LF environment navigable area is restricted to a gridworld setting with height and width of the grid within the discrete range $[4, 12]$.

The LF environment is structurally similar to the HF environment, but follows different dynamics. The agent's navigation actions are moving 1 cell forward if the cell ahead is clear or rotating $\pi/2$ clockwise or counter-clockwise. In the HF and LF target task, the agent receives a reward of $+1000$ upon crafting a stone-axe, and -1 for all other steps. In the LF environment, the agent's state representation is similar to the HF environment. The LF agent's sensor is identical to the HF agent, but it emits a beam at incremental angles of $\frac{\pi}{4}$ to account for the smaller OOMDP state space.

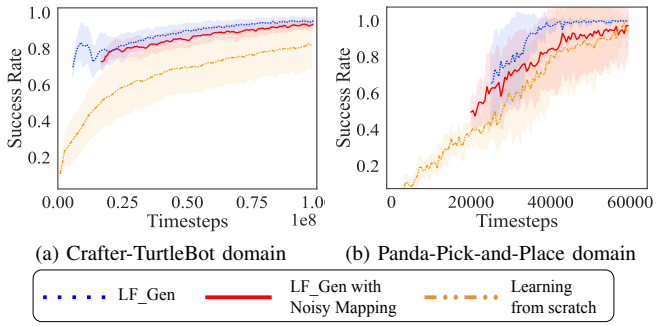
The Panda-Pick-and-Place domain (Fig 2b) involves a robotic arm picking up objects and placing them at their target locations [20], rendered in PyBullet [19]. The set of classes in the HF environment are $\mathcal{C}^{HF}_{man} = \{wall, objects\}$, and the class parameters are: $Par(wall) = \{length, width, height\}$, where these parameters can assume continuous values within $[0.05m, 0.4m]$; and parameter $Par(objects) = \{no\_of\_objects\}$; within a discrete range. The set of classes in the LF and HF environments are equivalent; $\mathcal{C}^{LF}_{man} = \{wall, objects\}$, but the range of values for all continuous parameters in the HF environment $(wall(length, width, height))$ are bounded within the set $\{2, ..., 8\}$, reducing the number of OOMDP states attainable.

The actions of the agent in the HF domain are continuous end-effector dispacements, while in the LF environment, the actions involve discrete end-effector displacements in the 3D gridworld. The agent's sensor provides the agent with its end-effector location along with the current location of the object and the target location, and the gripper state. The agent acts in a sparse reward setting, and receives a reward of $+1$ if it achieves the goal and -1 otherwise.

`LF_Gen` generates a LF target task from the HF target task[1]. The next step entails optimizing the curriculum for this LF target task using "ACuTE", and obtaining the HF curriculum using the mapping from `LF_Gen`[2].

[1]Code link: `https://github.com/shukla-yash/LF_Gen/`

[2]For curriculum optimization in LF, see Appendix Section A.
Link to Appendix: `https://github.com/shukla-yash/LF_Gen/blob/master/TechnicalAppendix.pdf`



(a) Crafter-TurtleBot domain (b) Panda-Pick-and-Place domain

Fig. 3: Learning Curves for Crafter-TurtleBot navigation domain (a) and Panda-Pick-and-Place manipulation domain (b).

Fig 3 shows the learning curves on the final HF target task with and without any curriculum (averaged over 10 trials). The learning curves for the curriculum approaches have an offset on x-axis to account for interactions in the source tasks of the curriculum, signifying a strong transfer [21]. From the learning curves, it is evident that the curriculum optimized using a low-fidelity (LF) environment helps in significantly reducing the number of interactions required to converge to a successful policy in the target task. The time-to-threshold between policy learned through curriculum and without any curriculum in the Crafter-TurtleBot domain is 5×10^7 timesteps and 2×10^4 in the Panda-Pick-and-Place manipulation domain. We also evaluate the curriculum schema transfer through imperfect mapping between the LF and the HF environments, by incorporating multivariate noise in the set of mapping functions. We observe that the schema of the curriculum transferred even with noisy mappings achieves improved time-to-threshold and better success rate performance than learning from scratch[3].

V. Conclusion and Future Work

We proposed an Object-Oriented framework for characterizing low-fidelity (LF) environments for complex high-fidelity (HF) environments. We demonstrated that object-oriented MDP provides a convenient way to generate a LF environment and obtain a mapping between the two environments. Our experiments show improved time-to-threshold performance on a robotic navigation and a robotic manipulation domain, even with imperfect mappings. Thus, `LF_Gen` can be employed in obtaining curriculum for complex robotic tasks, helping in scaling robot learning.

An extension of our approach will be to further automate the LF generation procedure. A potential solution is to utilize HF environment knowledge to come up with a mapping for the `LF_Gen` approach. Additionally, we would like to explore situations where a single robot performs navigation and manipulation behaviors. A possible approach will be to have separate LF environments for navigation and manipulation, interlinked to fulfill a common objective defined by the task in question.

[3]More details on experiments and results in Appendix Section B.
Link to Appendix: `https://github.com/shukla-yash/LF_Gen/blob/master/TechnicalAppendix.pdf`

REFERENCES

[1] Yifan Gao and Lezhou Wu. 2021. Efficiently Mastering the Game of NoGo with Deep Reinforcement Learning Supported by Domain Knowledge. Electronics 10, 13 (2021), 1533.

[2] Andrej Karpathy and Michiel Van De Panne. 2012. Curriculum learning for motor skills. In Canadian Conference on Artificial Intelligence. Springer, 325–330.

[3] Kirkpatrick, James, et al. "Overcoming catastrophic forgetting in neural networks." Proceedings of the national academy of sciences 114.13 (2017): 3521-3526.

[4] Atkinson, Craig, et al. "Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting." Neurocomputing 428 (2021): 291-307.

[5] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. 2020. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. J. of Machine Learning Research 21 (2020), 1–50.

[6] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In 2017 IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS). IEEE, 23–30.

[7] Höfer, S., Bekris, K., Handa, A., Gamboa, J.C., Golemo, F., Mozifian, M., Atkeson, C., Fox, D., Goldberg, K., Leonard, J. and Liu, C.K., 2020. Perspectives on sim2real transfer for robotics: A summary of the R: SS 2020 workshop. arXiv preprint arXiv:2012.03806.

[8] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Sim-to-real transfer of robotic control with dynamics randomization. In 2018 IEEE Intl. Conf. on robotics and automation (ICRA). IEEE, 1–8.

[9] Josip Josifovski, Mohammadhossein Malmir, Noah Klarmann, and Alois and Knoll. Continual Learning on Incremental Simulations for Real-World Robotic Manipulation Tasks. In 2nd Workshop on Closing the Reality Gap in Sim2Real Transfer for Robotics at Robotics: Science and Systems (R:SS) 2020.

[10] Paull, Liam, and Anthony Courchesne. On assessing the value of simulation for robotics. In 2nd Workshop on Closing the Reality Gap in Sim2Real Transfer for Robotics at Robotics: Science and Systems (R:SS) RSS. 2020.

[11] Shukla, Y., Thierauf, C., Hosseini, R., Tatiya, G., and Sinapov, J. (2022) ACuTE: Automatic Curriculum Transfer from Simple to Complex Environments. To appear in proceedings of the 2022 ACM Conference on Autonomous Agents and Multi-Agent Systems (AAMAS).

[12] Thomas Carr, Maria Chli, and George Vogiatzis. 2019. Domain Adaptation for Reinforcement Learning on the Atari. In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. 1859–1861.

[13] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. The International Journal of Robotics Research, 37(4-5):421–436, 2018.

[14] Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative Multi-agent Control Using Deep Reinforcement Learning. In Autonomous Agents and Multiagent Systems (Lecture Notes in Computer Science), Gita Sukthankar and Juan A. Rodriguez-Aguilar (Eds.). Springer International Publishing, Cham, 66–83. https://doi.org/10.1007/978-3-319-71682-45

[15] Sun, Charles, et al. "Fully Autonomous Real-World Reinforcement Learning with Applications to Mobile Manipulation." Conference on Robot Learning. PMLR, 2022.

[16] Christianos, Filippos, et al. "Scaling multi-agent reinforcement learning with selective parameter sharing." International Conference on Machine Learning. PMLR, 2021.

[17] Diuk, Carlos, Andre Cohen, and Michael L. Littman. "An object-oriented representation for efficient reinforcement learning." Proceedings of the 25th international conference on Machine learning. 2008.

[18] Silva, Felipe Leno Da, and Anna Helena Reali Costa. "Object-oriented curriculum generation for reinforcement learning." Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. 2018.

[19] Erwin Coumans and Yunfei Bai. 2016–2019. PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org.

[20] Gallouédec, Q., Cazin, N., Dellandréa, E., & Chen, L. (2021). panda-gym: Open-Source Goal-Conditioned Environments for Robotic Learning. In 4th Robot Learning Workshop: Self-Supervised and Lifelong Learning at NeurIPS.

[21] Matthew E Taylor and Peter Stone. 2009. Transfer learning for reinforcement learning domains: A survey. J. of Machine Learning Research 10, 7 (2009)