Learning Design and Construction with Varying-Sized Materials via Prioritized Memory Resets

Yunfei Li[1,♯], **Tao Kong**[2], **Lei Li**[3], **Yi Wu**[1,4,♮]
[1] Institute for Interdisciplinary Information Sciences, Tsinghua University
[2] ByteDance AI Lab, [3] University of California Santa Barbara, [4] Shanghai Qi Zhi Institute
♯`liyf20@mails.tsinghua.edu.cn`,♮`jxwuyi@mail.tsinghua.edu.cn`

Abstract

Can a robot autonomously learn to design and construct a bridge from varying-sized blocks without a blueprint? It is a challenging task with long horizon and sparse reward – the robot has to figure out physically stable design schemes and feasible actions to manipulate and transport blocks. Due to diverse block sizes, the state space and action trajectories are vast to explore. In this paper, we propose a hierarchical approach for this problem. It consists of a reinforcement-learning designer to propose high-level building instructions and a motion-planning-based action generator to manipulate blocks at the low level. For high-level learning, we develop a novel technique, *prioritized memory resetting* (PMR) to improve exploration. PMR adaptively resets the state to those most critical configurations from a replay buffer so that the robot can resume training on partial architectures instead of from scratch. Furthermore, we augment PMR with auxiliary training objectives and fine-tune the designer with the locomotion generator. Our experiments in simulation and on a real deployed robotic system demonstrate that it is able to effectively construct bridges with blocks of varying sizes at a high success rate. Demos can be found at https://sites.google.com/view/bridge-pmr.

1 Introduction

Reinforcement learning (RL) has been an increasingly promising paradigm for solving complex robotic manipulation tasks [14, 32, 1, 40], such as grasping [18], stacking [28, 23], object rearrangement [30], mobile manipulation [22, 39] and folding towels [4]. We tackle a challenging task, *bridge design and construction with varying-sized objects* (Fig. 1), where a collection of *varying-sized* materials are given while the robot needs to select necessary blocks to construct a stable bridge connecting two distant cliffs. In contrast with many existing manipulation tasks where the target object configuration is often known in advance [25, 43], neither the target bridge architecture nor the construction instructions are given in this task. Moreover, the success signal can be only obtained after a valid bridge is completely built. Hence, this problem is difficult for its *long horizon* and *sparse reward*, and requires non-trivial *exploration* due to varying block sizes and cliff width.

To tackle this challenge, we adopt a hierarchical solution consisting of an RL-based high-level designer and a planning-based low-level action generator. The high-level designer is trained by RL over object-centric states to decompose the long-horizon task into a sequence of single-block pick-and-place sub-tasks. The low-level policy simply executes the pick-and-place instructions by motion planning and produces a collision-free sequence of robot actions. This is conceptually similar to ReLMoGen [39]. ReLMoGen focuses on mobile robots and produces state-based sub-goals, such as spatial positions and arm states, while we consider object configurations. ReLMoGen trains

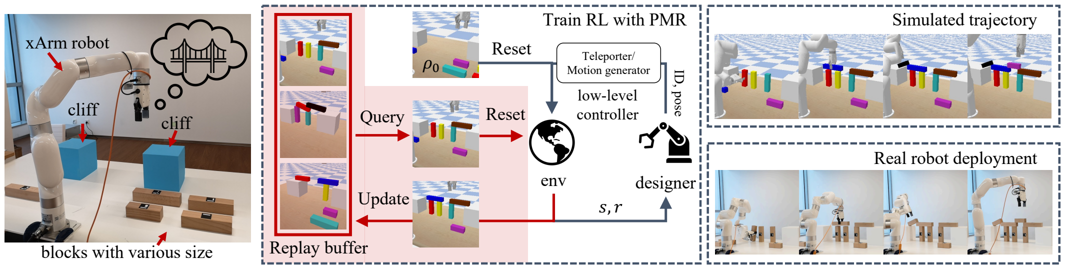


Figure 1: Bridge design and construction with a collection of varying-sized objects. *Left*: Task setting. *Middle*: Overview of training approach. An RL designer is trained with Prioritized Memory Resets (red shadowed area). The predicted instruction from designer is executed with a low-level controller. *Right*: Evaluation in simulation and on a real robot.

the high-level planner with a locomotion generator throughout the entire training process, which, however, is computationally expensive and results in poor exploration in our task. We propose to first train the high-level designer solely by teleporting the selected block to the target position and then fine-tune the designer with the action generator, which achieves a substantially higher success rate.

To overcome the exploration challenge when training the high-level designer, we propose a novel training paradigm, *Prioritized Memory Resets* (PMR). PMR adaptively resets the RL environment to a past state selected from the replay buffer, so that the robot can start from an intermediate half-done architecture instead of always restarting from scratch in each training episode. The insight of PMR is that in this sparse-reward hard-exploration problem, some close-to-success configurations may be hardly re-achieved from scratch by the training policy. Therefore, we directly reset the environment to those states that may lead to the biggest learning advancements. PMR is conceptually similar to automatic curriculum learning methods in goal-conditioned RL [12, 11], which trains the agent with adaptive *target goals* for fast policy improvements. By contrast, PMR resets the *initial state* of a training episode. We also adopt an auxiliary self-supervised objective for better representation learning, which further accelerates learning.

Experiment result shows that our RL-based bi-level solution achieves a success rate of 71.8% in simulation for constructing a bridge with 7 random-sized blocks and discovers interesting bridge architectures while the standard RL method fails completely. Ablation studies also demonstrate that all algorithmic components, including PMR, self-supervised learning, and locomotion fine-tuning, are critical to the overall performance. We also validate our method on a real-world robot arm.

2 Task setup

There are N building blocks on the table and two "cliffs" at a random distance from each other. The lengths of building blocks are sampled from three categories: standard length L which is equal to the cliff height, shorter length and longer length. A 7 DoF xArm robot is mounted on the side of the table, and aims to design and construct a bridge using the building blocks that can connect the cliffs. Each episode starts by sampling initial distribution ρ_0 where all the building blocks are aligned outside the valley. The agent observes all the objects in the scene , and instructs one building block to a new pose in each step. It only receives non-zero reward when a bridge is completely built. The agent is allowed 30 steps in each episode. Detailed description can be found in Appendix A.1.

We tackle this long-horizon manipulation task with a hierarchy of a high-level RL-based designer that sequentially instructs one object to a new pose, and a low-level controller that generates collision-free robot motions. The high-level RL designer is instantiated as an actor π_θ and a critic V_ϕ with a shared transformer-based [38] encoder ψ, and is trained with an algorithm built on Phasic Policy Gradient (PPG) [8]. The low-level controller is either an object teleporter that directly teleports the selected block to the target pose, or a sampling-based motion planner.

3 Method

Designing and constructing a bridge using varying-sized blocks is a challenging task due to sparse reward and complex physical constraints of stable bridges. Hence, we propose the PMR technique to tackle this hard exploration challenge. We further improve the representation learning of the agent with a self-supervised auxiliary task, which can accelerate training significantly. Finally, we fine-tune the pre-trained high-level designer with a motion generator to get feasible instructions for the robot.

Prioritized memory reset We allow the agent to restart from intermediate states it has previously visited with some probability instead of always from scratch when an episode resets. For on-policy RL methods, an agent can occasionally reach some promising states close to success by random exploration. However, the agent may not be able to re-visit them since each episode restarts from scratch in standard RL. By contrast, PMR enables the agent to directly teleport to these critical states without re-executing its policy.

We propose to use temporal difference (TD) error as a priority metric to select critical states: $\text{priority}(s) = |r + \gamma V(s') - V(s)|$, where s, r, s' are states, rewards, and the subsequent states collected from previous interactions with the environment, and V is the learned value function. Intuitively, the states that result in unexpected success or suddenly degrade to complete failure are with large TD errors. Restarting from these states could guide the agent towards successful configurations and practice to avoid catastrophic failure.

We store the visited states and their priorities when interacting with the environment. When an episode terminates, we query the state with the highest priority and set it as the new initial state with probability $p_{restart}$; otherwise, we reset the episode with a random state sampled from the initial state distribution ρ_0. Detailed implementation and pseudo code can be found in Appendix A.2.

Inverse dynamics prediction To better guide the training of the transformer-based encoder ψ, we propose to optimize a self-supervised auxiliary task jointly with the original RL objective. Given a transition tuple (s_t, a_t, s_{t+1}), the auxiliary task is to predict the action a_t that results in the transition from s_t to s_{t+1}, which is called inverse dynamics prediction. Any valid transitions from our environment can be used as training data for the auxiliary task. Therefore, the agent can always get rich supervision for learning representation even when the reward signal from the environment is very sparse. In our experiments, we optimize $\mathcal{L}^{joint} = \mathcal{L}^V + \beta_{clone}\mathcal{L}^{clone} + \beta_{aux}\mathcal{L}^{aux}$ in the value phase of PPG training. $\mathcal{L}^V$ and $\mathcal{L}^{clone}$ are the original objectives in PPG, and L^{aux} is the auxiliary prediction loss defined as the cross-entropy between predicted action $\hat{a}$ and ground truth a_t.

Fine-tuning with low-level control generator We first train the high-level designer with object teleportation, then fine-tune the learned designer by integrating with a motion generator. With an object teleporter, the simulator directly repositions the selected block to the instructed pose, runs simulation until all the objects become stable, then takes the resulting state as the next state. To avoid infeasible instructions for the robot, we further replace the object teleporter with a motion generator which fully simulates the arm movement, and fine-tune the pre-trained designer. Each instruction from the designer is implemented as a pick-and-place task that grasps the block's center of mass. We use bidirectional RRT [20] to search a collision-free sequence of robot motions. If the planner fails (which may due to failing to find a grasp pose, target state in collision, etc.), the simulator will revert the scene to the state before the whole pick-and-place task and wait for the next instruction.

Real robot deployment We mount an xArm7 robot in the real world with the same configuration as in simulation. A RealSense D435 RGBD camera is mounted on the hand of the robot. We attach ArUco markers [13] to the building blocks to get accurate pose estimation. We estimate the lengths of the building blocks using contour approximation provided in OpenCV [6]. After parsing the scene at the beginning of an episode, the robot plans a sequence of joint angle positions with the trained high-level designer and the low-level motion generator, then executes along the planned trajectory.

4 Experiments

Main results We demonstrate how our agent designs and constructs a long bridge with a total of 7 building blocks in Fig. 2. The building material set consists of three standard blocks with a length of 20cm, two short blocks of length 14cm, and two long blocks of length 24cm. The distance between cliffs is 65cm. The agent learns to put three standard blocks vertically inside the valley as supporting blocks, then put two long blocks on top of these supporting blocks as part of the bridge surface, finally fill in the gaps with two short blocks. More results can be found in Appendix B.

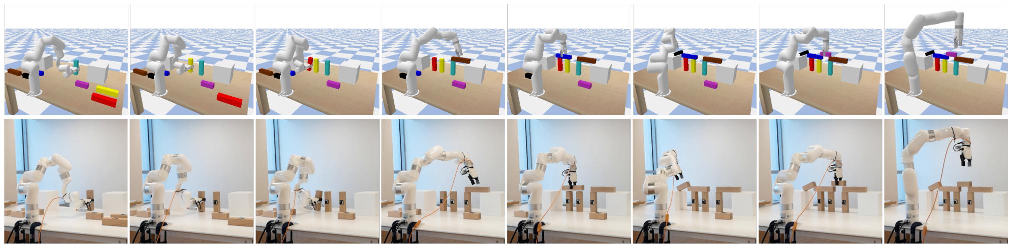


Figure 2: Learned strategies for constructing a long bridge using 7 blocks with different sizes. The two rows are construction sequences in simulation and in the real world.

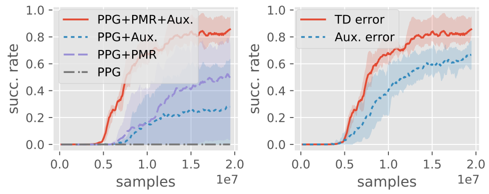


Figure 3: Ablation studies of different algorithm variants. The left figure shows the effectiveness of prioritized reset and auxiliary prediction task. Red: PPG with prioritized reset and auxiliary prediction task. Blue: PPG with auxiliary representation learning. Purple: PPG with prioritized reset. Grey: Pure PPG. The right figure compares the performances of different metrics to prioritized the reset states.

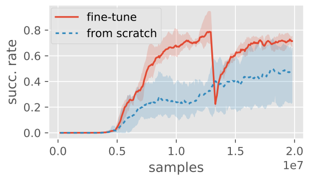


Figure 4: Comparison between pre-training high-level designer with teleportation then fine-tuning after integrated with a motion generator (red) and training from scratch combined with the low-level controller (blue). The first half of the red curve is evaluated with teleportation, and all other parts are evaluated with the motion generator.

Ablation studies on high-level designer learning We verify the effectiveness of PMR and auxiliary task by comparing our method with the variants that remove one or both components. All the experimented methods are evaluated on the tasks with a collection of 7 blocks of various sizes and are always reset from states sampled from ρ_0. The distance between cliffs is sampled from the range [2.75L, 3.75L]. In the left half of Fig. 3, our method ("PPG + PMR + Aux.", red) outperforms "PPG + Aux." (blue) and "PPG + PMR" (purple) with a large margin, indicating both PMR and auxiliary prediction are critical for efficient learning. Note that naively apply PPG algorithm, which is essentially the method in [24], leads to complete failure (grey). It also demonstrates this sparse-reward construction task with varying-sized building blocks is non-trivial for current on-policy RL algorithms. We compare different metrics to prioritize the reset states in the right half of Fig. 3. The red curve is prioritizing with absolute TD error, and the blue curve is with the inverse dynamics prediction error. Prioritizing with TD error achieves better sample efficiency and a higher success rate.

Ablation studies on fine-tuning with low-level control We take an RL designer trained with the object teleporter for $1.3e7$ timesteps, then continue training by combining it with the motion generator. As shown in Fig. 4, directly executing the instructions of the pre-trained policy with the motion generator can only achieve a success rate of 0.224, but the success rate can be improved to 0.718 after fine-tuning. We try another variant that trains the policy from scratch with mixed teleportation and motion generator. We use the motion generator to execute instructions with probability equal to the current success rate of the agent, and use the object teleporter otherwise. The overall sample efficiency is lower than pre-training then fine-tuning, and the success rate only converges to 0.472. The degradation of performance may due to insufficient exploration when integrated with the low-level controller in the early training stage.

5 Conclusion

We tackle a challenging sparse-reward manipulation task that designs and constructs bridges with varying-sized building blocks. We propose a novel learning paradigm PMR, that allows the agent to restart from critical states it has visited before to deal with the exploration issue. We additionally propose an auxiliary representation learning task and fine-tuning with integration of a motion generator to successfully build a system that can construct interesting structures using building blocks of various sizes in the real world.

References

[1] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 1726–1734. AAAI Press, 2017.

[3] Akhil Bagaria and George Konidaris. Option discovery using deep skill chaining. In *International Conference on Learning Representations*, 2020.

[4] Benjamin Balaguer and Stefano Carpin. Combining imitation and reinforcement learning to fold deformable planar objects. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011, San Francisco, CA, USA, September 25-30, 2011*, pages 1405–1412. IEEE, 2011.

[5] Dhruv Batra, Angel X. Chang, Sonia Chernova, Andrew J. Davison, Jia Deng, Vladlen Koltun, Sergey Levine, Jitendra Malik, Igor Mordatch, Roozbeh Mottaghi, Manolis Savva, and Hao Su. Rearrangement: A challenge for embodied AI. *CoRR*, abs/2011.01975, 2020.

[6] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[7] Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[8] Karl Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. *CoRR*, abs/2009.04416, 2020.

[9] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2020.

[10] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

[11] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning*, pages 1515–1528, 2018.

[12] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pages 482–495. PMLR, 2017.

[13] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[14] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[15] Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine. Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6244–6251. IEEE, 2018.

[16] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A. Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[17] Nicolas Heess, Gregory Wayne, Yuval Tassa, Timothy P. Lillicrap, Martin A. Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *CoRR*, abs/1610.05182, 2016.

[18] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[19] Ross A Knepper, Todd Layton, John Romanishin, and Daniela Rus. Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In *2013 IEEE International conference on robotics and automation*, pages 855–862. IEEE, 2013.

[20] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.

[21] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29:3675–3683, 2016.

[22] Chengshu Li, Fei Xia, Roberto Martín-Martín, and Silvio Savarese. HRL4IN: hierarchical reinforcement learning for interactive navigation with mobile manipulators. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pages 603–616. PMLR, 2019.

[23] Richard Li, Allan Jabri, Trevor Darrell, and Pulkit Agrawal. Towards practical multi-object manipulation using relational reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4051–4058. IEEE, 2020.

[24] Yunfei Li, Tao Kong, Lei Li, Yifeng Li, and Yi Wu. Learning to design and construct bridge without blueprint. *arXiv preprint arXiv:2108.02439*, 2021.

[25] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, A. M. Agogino, A. Tamar, and P. Abbeel. Reinforcement learning on variable impedance controller for high-precision robotic assembly. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3080–3087, 2019.

[26] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3307–3317, 2018.

[27] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[28] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pages 6292–6299. IEEE, 2018.

[29] L. Nägele, A. Hoffmann, A. Schierl, and W. Reif. Legobot: Automated planning for coordinated multi-robot assembly of lego structures*. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9088–9095, 2020.

[30] OpenAI OpenAI, Matthias Plappert, Raul Sampedro, Tao Xu, Ilge Akkaya, Vineet Kosaraju, Peter Welinder, Ruben D'Sa, Arthur Petron, Henrique P d O Pinto, et al. Asymmetric self-play for automatic goal discovery in robotic manipulation. *arXiv preprint arXiv:2101.04882*, 2021.

[31] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2778–2787. PMLR, 2017.

[32] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In Hadas Kress-Gazit, Siddhartha S. Srinivasa, Tom Howard, and Nikolay Atanasov, editors, *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*, 2018.

[33] Daniel Ritchie, Sharon Lin, Noah D Goodman, and Pat Hanrahan. Generating design suggestions under tight constraints with gradient-based probabilistic programming. In *Computer Graphics Forum*, volume 34, pages 515–526. Wiley Online Library, 2015.

[34] Daniel Ritchie, Ben Mildenhall, Noah D Goodman, and Pat Hanrahan. Controlling procedural modeling programs with stochastically-ordered sequential monte carlo. *ACM Transactions on Graphics (TOG)*, 34(4):1–11, 2015.

[35] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[36] Satinder Singh, Andrew G Barto, and Nuttapong Chentanez. Intrinsically motivated reinforcement learning. Technical report, MASSACHUSETTS UNIV AMHERST DEPT OF COMPUTER SCIENCE, 2005.

[37] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *International Conference on Learning Representations*, 2018.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[39] Fei Xia, Chengshu Li, Roberto Martın-Martın, Or Litany, Alexander Toshev, and Silvio Savarese. Relmogen: Integrating motion generation in reinforcement learning for mobile manipulation.

[40] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR, 2020.

[41] Kevin Zakka, Andy Zeng, Johnny Lee, and Shuran Song. Form2fit: Learning shape priors for generalizable assembly from disassembly. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9404–9410. IEEE, 2020.

[42] Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, 33, 2020.

[43] Yifeng Zhu, Jonathan Tremblay, Stan Birchfield, and Yuke Zhu. Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. *arXiv preprint arXiv:2012.07277*, 2020.

A Implementation details

We use Pybullet [9] to build the simulated environment.

A.1 MDP Formulation

The high-level bridge design problem is formulated as a Markov decision process defined as follows:

Observation: In each step, the agent observes the positions, orientations, velocities, and sizes of all the building blocks and cliffs. If a building block is outside of the valley, we replace its observation vector with a special token.

Action: The agent can instruct one building block to a new pose within the vertical plane that goes through the centers of cliffs. Each action is a vector of 4 elements $(\text{ID}, y, z, \text{angle})$, where ID denotes which block to move, y and z specify the target position of the block's center of mass, angle denotes the 1-D rotation within the plane.

Reward: For each step, the agent can get a 0.1 reward only if a bridge is successfully built; otherwise it receives no reward. We cast multiple rays downwards onto the structure inside the valley to detect its height. If the height of all the detected points is greater than the cliff height plus the block thickness, we consider the structure successful.

Horizon: Each episode lasts a fixed length of 30 steps.

Initial state distribution ρ_0: When the environment resets, the distance between the cliffs is sampled from $[0.75L, 3.75L]$. We then sample a set of building blocks consisting of $\lfloor N/2 \rfloor$ standard blocks, $\lceil N/4 \rceil$ long blocks, and $(\lceil N/2 \rceil - \lceil N/4 \rceil)$ short blocks. A long block is of length uniformly sampled from $1.1L$ to $1.25L$, and a short block is uniformly sampled from length $0.5L$ to $0.9L$. All the building blocks are aligned on the table outside the valley.

A.2 PMR implementation

We maintain all the visited states in a priority queue. Since our agent is constantly evolving, the stored priorities computed from old values would soon become stale. Therefore, we re-compute the priorities of all the tracked states after each training phase. We pop out the states with the least priorities when the priority queue is full. Since our state space is continuous and we can only restore a limited number of states, it is infeasible to keep track of all the visited states. Also, it is unnecessary to store multiple states that are similar to each other. Therefore, we adopt state hashing and only keep one representative for each hash value. The hash function we apply is defined as follows: we first get the heights of the built structure's upper surface, which we call the skyline vector, then discretize the vector as the hash key of the state.

The overall algorithm of RL with PMR is shown in Alg. 1.

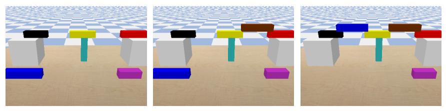
Algorithm 1: PPG with PMR

```
Intialize $\psi$, $\pi_\theta$, $V_\phi$. RL data buffer $\mathcal{B}$. An
  empty replay buffer $\mathcal{Q}$ to store reset states.
$s_0 \sim \rho_0$
for iter=0:n_iters do
    $\mathcal{B} \leftarrow \emptyset$
    for t=0:n_steps do
        $\mathcal{Q}$.insert(priority($s_t$), $s_t$)
        $a_t \leftarrow \pi_\theta(\psi(s_t))$
        $s_{t+1}, r_t$, terminate $\leftarrow$ env.step($a_t$)
        $\mathcal{B} \leftarrow$
          $\mathcal{B} \cup (s_t, a_t, r_t, s_{t+1}, \text{terminate})$
        if terminate then
            if $rand() < p_{restart}$ then
                $s_{t+1} \leftarrow \mathcal{Q}$.pop()
            else
                $s_{t+1} \leftarrow \rho_0$
    for batch_data sampled from $\mathcal{B}$ do
        Optimize $\psi$ and $\pi_\theta$ with $\mathcal{L}^\pi$
    for batch_data sampled from $\mathcal{B}$ do
        Optimize $\psi$ and $V_\phi$ with $\mathcal{L}^{joint}$
    Recompute priority($s$) for each $s$ in $\mathcal{Q}$
```

B More results

B.1 Visualization of Reset States with PMR

We visualize which states PMR proposes to restart from. The states with the highest priorities during training are demonstrated in Fig. 5. The quality of reset states is also evolving as the training proceeds. In the early stage, the states with large TD errors are messy states with objects randomly dropped in the scene. Then the agent gradually learns to construct more meaningful structures. Finally, the agent


(a) An efficient construction plan using 6 blocks.

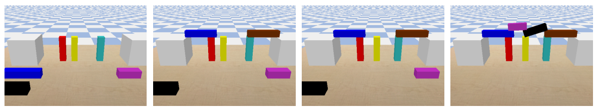

(b) Another bridge design using all 7 blocks.

Figure 6: Different modes of construction plans under the same task configuration.

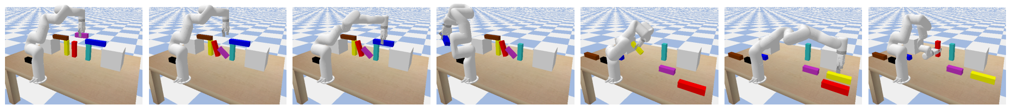


Figure 7: One failure case. The agent knocks down part of the built bridge when placing the short purple block, cleans up the messy scene by itself, and tries to build again. The trajectory terminates due to time limit.

focuses on building very long bridges from partially built structures. The prioritized reset mechanism can be also viewed as an implicit curriculum for the agent.

B.2 Learned strategies

Our agent can discover multiple solutions for the same task configuration. We set the distance between cliffs to be 69cm, and give the agent 7 blocks of length 14cm, 18cm, 20cm, 20cm, 20cm, 24cm, 24cm. In Fig. 6, the agent discovers two different construction plans. The first solution only uses 6 blocks to solve the task. In the second solution, the agent utilizes all 7 blocks. It strategically adjusts the position of the yellow block before putting the short black block.

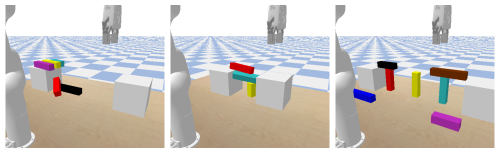

(a) Iteration 10 (b) Iteration 50 (c) Iteration 100

Figure 5: Reset states with top priorities selected by PMR at different training iterations.

B.3 Failure cases

A failure case of the agent is depicted in Fig. 7. The agent intends to connect the red and light blue blocks with the short purple block, but knocks down other blocks when dropping the pink block from the air. The agent then spends many steps to clear the scene, and tries to construct again. The agent fails to reach a successful state before the episode terminates.

C Related work

Robot construction and manipulation tasks [19, 29, 41, 15, 23, 43, 5] serve as a popular testbed for developing intelligent manipulators with long-term autonomy. Most of the construction tasks assume a known target state in a priori, i.e., the desired configuration designed by a human expert is provided to the robot. We focus on a bridge design and construction task with no prior knowledge of the precise target state. So, the robot has to both design the bridge architecture and construct the bridge via a sequence of feasible control actions. There are also works focusing on generating structure designs under particular constraints without considering construction [33, 34].

Hierarchical frameworks are commonly adopted in complex long-horizon manipulation tasks. Hierarchical reinforcement learning (HRL) typically learns a bi-level policy, with the high-level policy generating sub-goals for the low-level policy to execute [21, 2]. The two policies can be optimized jointly [26, 27, 3] or separately [17]. ReLMoGen [39] tackles mobile manipulation and interactive navigation tasks with a combination of a learnable high-level policy and a fixed low-level motion generator. Our framework is conceptually similar to ReLMoGen, with a high-level RL designer integrated with a classical motion generator while our high-level policy considers object configurations instead of spatial positions or robot states. Moreover, we only use locomotion to fine-tune the high-level policy for better designer exploration while ReLMoGen leverages both parts throughout training.

The proposed prioritized memory reset technique adaptively proposes critical intermediate states for the agent to restart from, which is related to automatic curriculum learning methods that propose tasks with moderate difficulty [42, 12, 11, 37]. However, these methods mainly work for goal-conditioned problems or a fixed set of tasks by generating goals, while we directly reset the environment to previously visited states. Also, the curriculum learning methods in goal-conditioned RL typically assume a known goal space, while PMR does not need to explicitly know the space of state configurations. PMR enhances exploration by directly teleporting the agent to previous states without changing the reward function, which is different from classical intrinsic-reward-based exploration [36, 7]. PMR is most related to Go-Explore [10], which also teleports the agent to promising past states. However, Go-Explore uses a count-based metric as its state selection criterion, which exhaustively explores the entire state space. This is infeasible in complex construction tasks where exponentially many failure/unstable states exist and only a few architectures are crucial for success. Hence, PMR adopts a value-error-based criterion, which gradually learns to only focus on stable states as training proceeds. We remark that some model-based RL methods [35] also restore a visited state to perform monte-carlo tree search. However, these methods require an accurate forward model while PMR is model-free. Finally, we use inverse dynamics prediction as an auxiliary task for better representation learning. This self-supervised objective has been also applied in other problems including exploration [31] and meta-learning [16].