An Open-source Recipe for Building Simulated Robot Manipulation Benchmarks

Jiayuan Gu[1], Linghao Chen[2], Zhiwei Jia[1], Fanbo Xiang[1], Hao Su[1]
University of California San Diego[1], Zhejiang University[2]

***Abstract*—Benchmarking robot manipulation in simulation presents unique challenges. We emphasize physically realistic benchmarks that can be used to measure the progress of robot manipulation. We provide an open-source recipe for building simulated robot manipulation benchmarks, through the case study of ManiSkill2, a new benchmark suite for generalizable robot manipulation. Besides, we share our experience of sim-to-real experiments on the tasks of ManiSkill2. The goal of this paper is to promote open discussion and collaboration on building high-quality benchmarks for Embodied AI research.**

I. Introduction

Recent impressive progress in large language models (LLM) trained on internet-scale data like OpenAI GPT series [1] is considered a milestone for AI. In robotics, Google has proposed SayCan [2], RT-1 [3] and PaLM-E [4], which empower large language models with embodiment to tackle daily chores. However, many recent advancements tend to be "closed-source", as demanding computation resources and high-quality data are not accessible to the open-source community. For instance, SayCan [2] uses *"68000 teleoperated demonstrations that were collected over the course of 11 months using a fleet of 10 robots"* for behavior cloning. Collecting data and evaluating algorithms in the real world for robotics, especially robot manipulation, can be costly and hard to reproduce. Thus, simulation is an alternative. It is more affordable, with no cost to maintain and reset robots; it is also more accessible, as people can run simulations on devices like consumer-level laptops or cloud clusters. Moreover, it facilitates standardized evaluation protocols, as it is easier to ensure the same hardware and environment setups, as well as to compute metrics in simulation. Therefore, prototyping in simulation before deployment in the real world has become a common practice in robotics.

Nevertheless, benchmarking robot manipulation in simulation poses unique challenges. *First, a consensus on standardized tasks to measure the progress of robot manipulation has yet to be reached.* In contrast, standardized tasks like image classification and machine translation have catalyzed computer vision and natural language processing. Rearrangement [5] has been recently proposed as a canonical task for Embodied AI. The goal is to bring a given physical environment into a specified state. However, the instantiation of rearrangement in simulation can vary widely and have an impact on the quality of a benchmark. For example, quite a few benchmarks [6], [7] adopt abstracted grasping, and consequently, grasping is almost irrelevant to the physical properties of an object, which obviously results in sim-to-real gaps. *Besides, building simulated environments with a reasonable abstraction of the real world is non-trivial.* Creating physically realistic benchmarks like [8], [9] demands a systematic effort and interdisciplinary expertise. The devil is in the details. Unfortunately, there is a lack of open-source solutions to reveal the design choices, criteria and workflows during development. *Furthermore, a holistic understanding of sim-to-real gaps is underexplored*, whereas many efforts such as [10] have been made to study specific aspects. A high-quality simulated benchmark, which is intended to serve as a proxy of the real world, should ensure that sim-to-real gaps can be narrowed with minimal effort, so that the progress made in simulated benchmarks is substantial.

In this paper, we disclose the workflow and design choices during the development of our recent work ManiSkill2 [11], as a case study of building robot manipulation benchmarks in simulation. In addition, we share our hands-on experience of sim-to-real experiments on some tasks [12] of ManiSkill2, to provide insight into possible sources of and solutions to sim-to-real gaps. We hope this paper can serve as an open-source recipe for future works on building simulated robot manipulation benchmarks with minimal sim-to-real gaps.

II. A Case Study of Building Robot Manipulation Benchmarks in Simulation: ManiSkill2

In this section, we use our recent work ManiSkill2 [11] to present a case study of building robot manipulation benchmarks in simulation. ManiSkill2 is a unified benchmark for learning generalizable robot manipulation skills powered by SAPIEN [13]. It features 20 out-of-box task families with 2000+ diverse object models and 4M+ demonstration frames. Moreover, it empowers fast visual input learning algorithms so that a CNN-based policy can collect samples at about 2000 FPS with 1 GPU and 16 processes on a workstation. The benchmark can be used to study a wide range of algorithms: 2D & 3D vision-based reinforcement learning, imitation learning, sense-plan-act, etc.

Our workflow to build ManiSkill2 highlights a *verification-driven iterative development process*. It consists of 3 stages to create a task: task creation, reward design, and observation design. We verify the task via different approaches at each stage and may return to the previous stage if the verification fails. Additionally, the verification process also results in the acquisition of demonstrations. Fig 1 illustrates the workflow.

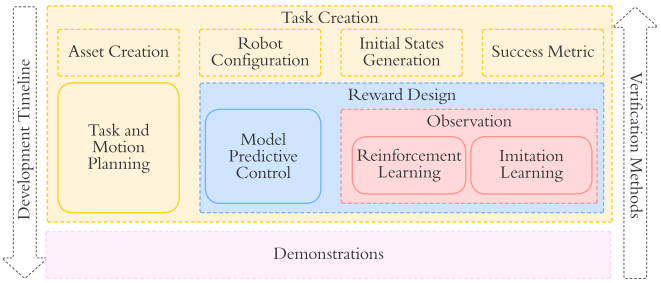

Fig. 1. The workflow to build ManiSkill2 [11].

A. Task Creation

The first stage (*task creation*) focuses on building task essentials, including creating assets, configuring robots, generating initial states, and defining success metrics.

To ensure task feasibility, we initially attempt to solve the task by manually controlling the robot end-effector through keyboards. However, this approach is only viable for a limited number of tasks, and mainly serves to locate trivial errors (such as objects that are too heavy to pick up). Teleoperation through more advanced devices like SpaceMouse or VR might help. Subsequently, we try to program task and motion planning (TAMP) solutions given full knowledge of the simulated environments. TAMP is free of crafting reward, and is suitable for many stationary manipulation tasks like pick-and-place, but shows difficulty when tackling underactuated systems (e.g. pushing chairs and moving buckets in [9]).

a) Assets: There are multiple public datasets of rigid objects [14]–[16], articulated objects [13], [17], and static scenes [18], [19]. Moreover, realistic layouts [6], [7], [20] of objects in scenes are also a type of asset. In general, objects used in simulation need to support two formats: collision (for simulation) and visual (for rendering). Note that collision and visual meshes can be quite different. For most rigid-body simulators, a convex shape is required for collision detection. Thus, convex decomposition (e.g., [21] used in ManiSkill2) is applied to approximate a given shape by a set of convex shapes. The hyper-parameters need to be tuned to adjust the number of convex shapes, which is a trade-off between higher fidelity and faster simulation. In addition, it is more tricky for articulated objects, as links may need to be processed separately with different hyper-parameters. For example, for a task like opening cabinet doors, handles should include more details while other parts can be modeled coarsely. For rendering, a visual shape is required, and not necessary to be convex. But mesh simplification is sometimes needed to speed up rendering. In order to enhance photorealism, it is important to tune the materials of objects and the lighting. Note that static scenes (e.g., [18], [19]) reconstructed from scans can look unrealistic when CAD objects are placed in them.

Furthermore, physical properties (e.g., scale, mass, inertia, friction) are also important but rarely provided in public datasets. Especially, grasping is affected by both the scale of the object and the width of the gripper. We tune those properties through trial and error. For example, if the object can not be grasped, we might adjust its scale. If it can be grasped but not lifted, we will adjust the density.

b) Robot configuration: The robot configuration involves configuring controllers and physical properties. The controller maps actions from the policy to motor signals that drive the robot. Thus, the controller decides the action space. Controllers mainly differ in input actions. For instance, the end-effector-space (task-space) controller takes the target end-effector pose as input. It is suitable for pick-and-place tasks [22], but can be insufficient to avoid obstacles. In contrast, the joint-space controller, which takes the target joint positions as input, has full control of the robot, while can be more challenging for learning algorithms [11]. Motion planning requires a joint-space controller. ManiSkill2 provides a wide range of controllers, and scripts to convert the action sequence for one controller to a sequence for the desired one. It enables us to make full use of demonstrations collected by different controllers.

A basic fact, but sometimes overlooked by who are not familiar with robotics, is that it always takes time for a controller to achieve the target. In ManiSkill2 and many other benchmarks like [6], [7], the action is first converted to the target joint positions, which are input to the underlying PD controller, which actually outputs motor signals (torque or force). The PD controller has two parameters: stiffness (P) and damping (D). In short, the larger the stiffness (or damping), the larger the force or torque is generated to reduce the error between the current and target positions (or velocities). These parameters need to be tuned to ensure natural and realistic movement of the robot. For example, if stiffness is too large, the robot can reach its target fast but may vibrate around the target.

c) Initial states: A task is characterized by its initial states, such as the initial joint positions of the robot and object poses. The difficulty to learn a policy is significantly affected by the diversity of its initial states. Generating initial states for some tasks, such as pick-and-place, can be done using simple heuristics, such as randomly sampling object poses within the workspace. However, for other tasks, such as the *AvoidObstacles* task in ManiSkill2, we adopt rejection sampling. In this case, we randomly sample the poses of obstacles and use motion planning to check if there exists a feasible solution.

d) Success metric: The success metric defines the terminal states of a task and its difficulty. In ManiSkill2, we have designed the success metric to not only consider the achievement of the task objective but also to demand the robot to keep stationary when the task is successfully completed. Note that there are also hyperparameters in success metrics, such as the distance threshold between the object and the goal to determine the success of a pick-and-place task.

B. Reward Design

The second stage aims at prototyping shaped reward functions. The reward function is a requisite for many methods like

Model Predictive Control (MPC) and Reinforcement Learning (RL). MPC is able to search solutions to difficult tasks given well-designed shaped rewards without training or observations. However, it is relatively slow since each episode is handled individually without knowledge sharing. RL requires additional training and hyperparameter tuning, but is more scalable than MPC during inference.

C. Observation Design

The third stage addresses observation spaces. For state observations, we in general includes the poses of all task-relevant objects and proprioception information (joint positions and velocities). Besides, the end-effector pose is also included to ease learning. For visual observations, camera parameters and placements need to be tailored for tasks so that visual observations contain adequate information. To verify this stage, we train state-based or visual-based RL policies and adjust the observations based on the performance of the trained policies.

D. Cloud Based Evaluation System

We build a cloud-based evaluation system based on Kubernetes, to benchmark different algorithms. Users can register accounts and submit their solutions. The solutions are in the form of docker images. The user needs to implement a function that accepts observations and returns actions. When a solution is submitted, the system will pull the its docker image and evaluate the included solution. The results are uploaded to a database and displayed on a public leaderboard.

III. Sim-to-real Experiments on ManiSkill2

In this section, we share our hands-on experience of sim-to-real experiments for the recently proposed CoTPC [12]. We focus on two ManiSkill2 tasks, *PickCube* and *PegInsertionSide*. In *PickCube*, the policy needs to control the robot arm to pick up a cube and move it to the goal position. In *PegInsertionSide*, the policy needs to insert a peg into the horizontal hole in a box. We use a 7-DoF xArm robotic arm with a two-finger gripper in the experiments. In the simulation setup, we train a state-based policy by behavior cloning on demonstrations collected by scripted motion planning solutions. In the real-world setup, a fixed RealSense D435 camera is used to acquire observations for state estimation. Fig 2 illustrates our simulated and real-world setups. Videos can be found on the project website (https://zjia.eng.ucsd.edu/cotpc).

We list the sources of sim-to-real gaps in our experiments. Surprisingly, we manage to sidestep difficulties through multiple workarounds. We hope our results can motivate new tasks that can better examine the sim-to-real transferability of algorithms.

A. State Estimation

It is common to train a policy that takes states as inputs rather than raw visual observations since state-based policies are easier and faster to acquire. [23] even shows that a transferable policy for in-hand manipulation can be learned using only proprioception history. However, if privileged states (e.g., object poses) are included in the input to the policy, they need to be estimated during real-world deployment. State estimation introduces noise (estimation error) and can not guarantee temporal consistency. In addition, its accuracy can be affected by multiple factors such as lighting and occlusion. A common solution is to introduce extensive data augmentation [24] during training in simulation. But data augmentation tends to smooth out the policy, which can be fatal to precise manipulation like *PegInsertionSide*. We argue that it can be critical to include observation history, which enables the robot to adapt to noisy input states on the fly.

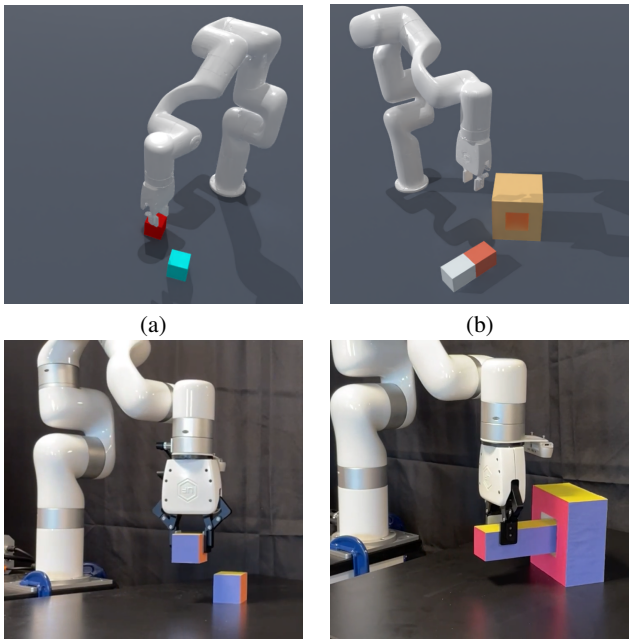

Fig. 2. **Simulation and real-world experiments setups.** (a) and (b) are the setups in simulation, and (c) and (d) are the setups in real-world experiments.

In the simulation setup, our policy, based on CoTPC [12], takes (state) observation and proprioception history as inputs. The states are obtained from the simulator, and no data augmentation is applied. In the real-world setup, we first estimate the 6-DoF object poses in the camera space by PVNet [25] and then transform the object poses to the robot base using the relative pose between the camera and the base of the robot arm obtained by hand-eye-calibration. To reduce the estimation errors, we only estimate the object poses at certain key-frames. Concretely, we estimate the object poses at the initial step and when the object is first grasped. For the succeeding steps, we can compute the relative pose between the object and the end-effector and then update the object pose according to both the acquired relative pose and the current end-effector pose. Note that such a workaround shares the same quasi-static assumption as motion planning: the grasped object is relatively fixed to the end-effector and there is no disturbance to the object poses.

B. Controller

The controller is the interface between the policy and the robot hardware. It converts the output action of a policy to motor signals that drive the robot. Without loss of generality, we take the position-based PD controller as an example. Its input is the target joint positions (and the target joint velocity is always zero). The action of our policy is the difference between the desired target joint positions and the current ones. The sim-to-real gap in the controller mainly results from the underlying implementation of the PD controller. Such implementation varies across different robots and is usually closed-source. The common practice is to tune the parameters or modify the algorithm of controllers in simulation, to align the real-world behaviors of robots.

In both simulation and real-world setups, the controller receives actions at a frequency of 5Hz. Note that it is different from the frequency (20Hz) used in the original version in ManiSkill2. The reason to use a lower frequency is that it is much easier to align kinematic behaviors than dynamic ones. When the control frequency is low, the controller can achieve the target position at each step without much parameter tuning, in both the simulation and real world. It is sufficient to align kinematic behaviors only for quasi-static manipulation.

C. Latency

The simulated environment usually follows the OpenAI Gym interface [26], which is synchronous. Specifically, at each timestep, there is no latency between events: the observations are perceived, the policy outputs the action given the observations at the same timestep, and the action is input to the controller and executed. However, latency is inevitable in the real world. It takes time to transfer observations and control signals between the robot and the host computer. Besides, latency can also result from state estimation and policy inference, especially when neural networks are included. Therefore, most robotic applications follow the asynchronous framework ROS. One existing solution is to train the policy with lagging observations in simulation, while at the cost of degraded performance [7]. Nonetheless, since the states are only estimated at key frames and the tasks are quasi-static, we sidestep this gap.

D. Physical Properties

It is difficult to measure all physical properties (e.g., inertia, friction coefficient) of an object precisely. Even if the measurement is precise enough, modeling physical properties in simulation to exactly match the real world is still troublesome. Thus, the gap in physics is almost inevitable. There are two common solutions: system identification and domain randomization [10]. System identification builds a model of the target but unknown dynamic system using measurements of the input and output signals of the system. The estimated model is able to support model-based approaches (e.g., model predictive control). Domain randomization randomizes the dynamics of the environment during policy training, which enables successful sim-to-real transfer [27]. However, most pick-and-place tasks might not need special attention to physical properties, since the force-closure grasp is quite robust to mild noise of physical properties. We do not handle this gap in our experiments.

IV. Conclusion

In this paper, we present the development of ManiSkill2, a new simulated robot manipulation benchmark. Besides, we discuss our sim-to-real experiments for CoTPC on two tasks of ManiSkill2. We hope our results of sim-to-real experiments can encourage the community to rethink whether existing works can only address quasi-static manipulation, and to explore approaches to narrow sim-to-real gaps related to dynamics. We hope that this paper will serve as a useful resource for researchers interested in building simulated robot manipulation benchmarks that are useful for real-world robotic applications. By continuing to investigate and address sim-to-real gaps, we can enable more efficient and effective deployment of robotic systems in the real world.

REFERENCES

[1] 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.

[2] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[3] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[4] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu *et al.*, "Palm-e: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[5] D. Batra, A. X. Chang, S. Chernova, A. J. Davison, J. Deng, V. Koltun, S. Levine, J. Malik, I. Mordatch, R. Mottaghi *et al.*, "Rearrangement: A challenge for embodied ai," *arXiv preprint arXiv:2011.01975*, 2020.

[6] K. Ehsani, W. Han, A. Herrasti, E. VanderBilt, L. Weihs, E. Kolve, A. Kembhavi, and R. Mottaghi, "Manipulathor: A framework for visual object manipulation," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4495–4504, 2021.

[7] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets *et al.*, "Habitat 2.0: Training home assistants to rearrange their habitat," *Advances in Neural Information Processing Systems*, vol. 34, pp. 251–266, 2021.

[8] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín, "robosuite: A modular simulation framework and benchmark for robot learning," *arXiv preprint arXiv:2009.12293*, 2020.

[9] T. Mu, Z. Ling, F. Xiang, D. C. Yang, X. Li, S. Tao, Z. Huang, Z. Jia, and H. Su, "Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[10] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[11] J. Gu, F. Xiang, Z. Ling, X. Wei, X. Liu, X. Li, R. Chen, S. Tao, T. Mu, P. Xie *et al.*, "Maniskill2: A unified benchmark for generalizable manipulation skills," in *International Conference on Learning Representations*.

[12] Z. Jia, F. Liu, V. Thumuluri, L. Chen, Z. Huang, and H. Su, "Chain-of-thought predictive control with behavior cloning," in *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*.

[13] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang *et al.*, "Sapien: A simulated part-based interactive environment," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 097–11 107.

[14] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *2015 international conference on advanced robotics (ICAR)*. IEEE, 2015, pp. 510–517.

[15] D. Morrison, P. Corke, and J. Leitner, "Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4368–4375, 2020.

[16] H.-S. Fang, C. Wang, M. Gou, and C. Lu, "Graspnet-1billion: A large-scale benchmark for general object grasping," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 444–11 453.

[17] L. Liu, W. Xu, H. Fu, S. Qian, Q. Yu, Y. Han, and C. Lu, "Akb-48: a real-world articulated object knowledge base," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 809–14 818.

[18] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson Env: real-world perception for embodied agents," in *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

[19] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma *et al.*, "The replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[20] C. Li, C. Gokmen, G. Levine, R. Martín-Martín, S. Srivastava, C. Wang, J. Wong, R. Zhang, M. Lingelbach, J. Sun *et al.*, "Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation," in *6th Annual Conference on Robot Learning*, 2022.

[21] X. Wei, M. Liu, Z. Ling, and H. Su, "Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–18, 2022.

[22] R. Martín-Martín, M. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space. an action space for reinforcement learning in contact rich tasks," in *Proceedings of the International Conference of Intelligent Robots and Systems (IROS)*, 2019.

[23] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik, "In-Hand Object Rotation via Rapid Motor Adaptation," in *Conference on Robot Learning (CoRL)*, 2022.

[24] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. Sundaralingam *et al.*, "Dextreme: Transfer of agile in-hand manipulation from simulation to reality," *arXiv preprint arXiv:2210.13702*, 2022.

[25] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4561–4570.

[26] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[27] X. Chen, J. Hu, C. Jin, L. Li, and L. Wang, "Understanding domain randomization for sim-to-real transfer," *arXiv preprint arXiv:2110.03239*, 2021.