AN EMPIRICAL STUDY AND ANALYSIS OF LEARNING GENERALIZABLE MANIPULATION SKILL IN THE SAPIEN SIMULATOR

Kun Liu[1], **Huiyuan Fu**[1], **Zheng Zhang**[1], **Huanpu Yin**[2]
Beijing University of Posts and Telecommunications[1]
Beijing Technology and Business University[2]
{liu_kun,fhy,zhangzheng}@bupt.edu.cn,yinhuanpu@btbu.edu.cn

ABSTRACT

This paper provides a brief overview of our submission to the no interaction track of SAPIEN ManiSkill Challenge 2021. Our approach follows an end-to-end pipeline which mainly consists of two steps: we first extract the point cloud features of multiple objects; then we adopt these features to predict the action score of the robot simulators through a deep and wide transformer-based network. More specially, to open up avenues for exploitation of learning manipulation skill, we present an empirical study that includes a bag of tricks and abortive attempts. Finally, our method achieves a promising ranking on the leaderboard. All code of our solution is available at https://github.com/liu666666/bigfish_codes.

1 INTRODUCTION

We perceive the real world by seeing the 3D environments and manipulating 3D objects surrounding us. Nowadays, researchers in the computer vision and deep learning community pay more and more attention to establishing artificial intelligence that can recognize and interact with objects in 3D scenes as human beings do. To this end, SAPIEN Open-Source Manipulation Skill Challenge, termed as ManiSkill Challenge Mu et al. (2021), was introduced to concentrate on learning unseen objects manipulation skills.

Based on the SAPIEN Manipulation Skill Benchmark (ManiSkill Benchmark) Xiang et al. (2020), ManiSkill challenge utilizes a variety of articulated objects in a full-physics simulator to accomplish 4 tasks: Opening Cabinet Door, Opening Cabinet Drawer, Pushing Chair, and Moving Bucket. Figure 1 demonstrates four tasks according to the examples from ManiSkill Benchmark, which consists of 162 objects and 36,000 successful demonstrations in total. It is noted that these tasks require manipulating the unseen test objects after learning on training objects where the training and testing samples are from the same class.

To learn the generalizable manipulation skill, we propose an end-to-end framework that extracts the 3D visual features and predicts the action of the robot arms. More specially, we adopt the PointNet Qi et al. (2017) to learn the point cloud features and then utilize a transformer Vaswani et al. (2017) based network to output the action of robots. Moreover, we provide an empirical study that is mainly composed of useful tricks and failure trials to learn generalizable manipulation skills.

2 RELATED WORK

To clarify, we classified the related works into two aspects: 3D simulators and point cloud feature extractors.

At present, more and more 3D simulators are constructed for Embodied AI. AI2-THOR Kolve et al. (2017) is a platform that provides the manipulation capabilities of agents, especially the interaction with actionable objects in four room categories: bed, bathroom, kitchen, and living room. The Interactive Gibson Environment (iGibson) Xia et al. (2020) can render dynamical environments by using photogrammetry for their room construction. Besides, iGibson provides users with ten

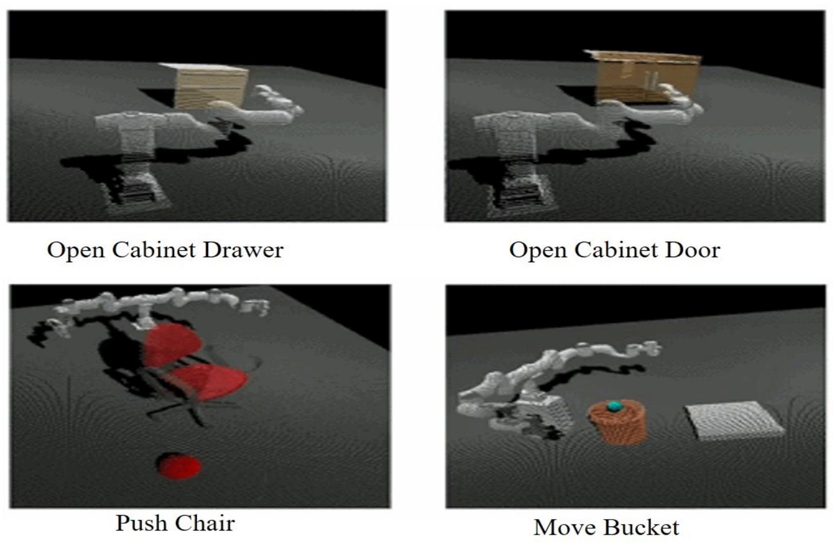

Figure 1: The samples of four manipulation tasks: Open Cabinet Door, Open Cabinet Drawer, Push Chair, and Move Bucket

fully functional and common robotic agents, which can improve the application in the real world. This challenge adopts the SAPIEN environment that supplies the interaction with diverse articulated objects without any commercial software. More details about 3D simulators can be found in this survey Duan et al. (2022).

This challenge simulates a moving panoramic camera that provides ego-centric point clouds or RGB-D images. Following Mu et al. (2021), we adopt the point clouds as the 3D visual input and utilize the PointNet Qi et al. (2017) as the feature extractor. PointNet extracts the powerful point cloud feature by learning pointwise features independently through several MLP layers and modeling global features via a max-pooling layer. PointNet ++ Charles et al. (2017) improves PointNet by designing a hierarchical network to model fine geometric architecture from the neighborhood of each point. Furthermore, Structural Relational Network (SRN) was proposed in Duan et al. (2019) to learn structural relational information between different local units using MLP. More details about point cloud can be referred to this paper Guo et al. (2020).

3 Method

In this section, we introduce our method briefly. Basically, we adopt an end-to-end architecture to extract the point cloud features and predict the action of robot arms. Specifically, we utilize the PointNet to extract the 3D feature of manipulated objects and develop a deep and wide transformer-based network to output the direction and the moving distance of the arm simulators.

4 Experiments

To give guidance for future work, we present an empirical study that includes a bag of tricks and abortive attempts. Specially, we tried various combinations of training samples with different trajectories, different head numbers in transformer architecture, the depth of transformer network, batch size, iteration number, and so on.

We report the results of four tasks on Table 1, Table 2, Table 3 and Table 4, respectively. We can observe that the opening cabinet drawer and opening cabinet door is easier than moving a bucket and pushing the chair. Specially, the former tasks achieve more than 0.4 success rate while the latter tasks obtain less than 0.3 success rate. To our surprise, models that perform close on the training set can differ by more than 10% points on the testing set, such as row 3 and row 4 in Table 1. Besides,

Table 1: The submission result on Opening Cabinet Drawer task

Row	Training Success Rate	Testing Success Rate	Detail
1	0.37	0.245	6 Block, 4 Head, 150K Iteration, 128 Batch Size(baseline)
2	0.47	0.336	Same as above besides 300K Iteration, 1024 Batch Size
3	0.69	0.404	6 Block, 64 Head, 900K Iteration, 1024 Batch Size
4	**0.70**	**0.508**	**12 Block, 64 Head, 900K Iteration, 1024 Batch Size**
5	0.70	0.496	Just submit above model again
6	0.68	0.444	16 Block, 16 Head, 900K Iteration, 1024 Batch Size

Table 2: The submission result on Opening Cabinet Door task

Row	Training Success Rate	Testing Success Rate	Detail
1	0.30	0.205	6 Block, 4 Head, 150K Iteration, 128 Batch Size(baseline)
2	0.34	0.244	12 Block, 16 Head, 300K Iteration
3	0.53	0.384	12 Block, 16 Head, 900K Iteration
4	0.56	0.320	Same as above but trained with various trajectories
5	0.52	0.280	12 Block, 64 Head, 900K Iteration
6	0.58	0.388	16 Block, 16 Head, 900K Iteration
7	**0.60**	**0.416**	**Fine-tuned on above best model**

as shown in row 4 and row 5 in Table 1, submitting the same model can achieve a 1.2% increment. Next, we summary up some advice from two aspects: useful tricks and failed attempts.

In this section, we list a bag of useful tricks.

1. Compared with baseline Mu et al. (2021), the model trained with a larger batch size and longer iteration number can bring significant performance improvement. Taking row 1 and row 2 in Table 1 as an example, larger batch size and longer iteration number results in 9.1% absolute increment from 24.5% to 33.6% on the Opening Cabinet Drawer task. Consequently, we adopt these hyper-parameters settings for the following experiments.
2. Compared with the model trained with suitable iteration and batch size, to some degree, deeper (more blocks) and wider (more heads) transformer networks increase the success rate drastically. Taking row 2 and row 3 in Table 1 as an example, wider networks, e.g., from 4 heads to 64 heads, bring 6.8% improvement in terms of testing success rate. As for deeper networks, Taking row 3 and row 4 in Table 1 as an example, 12 blocks outperform 6 blocks significantly (10.4%) on the testing set.

Table 3: The submission result on Moving Bucket task

Row	Training Success Rate	Testing Success Rate	Detail
1	0.15	0.115	6 Block, 4 Head, 150K Iteration, 128 Batch Size(baseline)
2	0.24	0.280	6 Block, 16 Head, 900K Iteration, 1024 Batch Size
3	0.30	0.292	12 Block, 16 Head, 900K Iteration, 1024 Batch Size
4	0.31	0.280	6 Block, 64 Head, 900K Iteration, 1024 Batch Size
5	0.35	**0.372**	**12 Block, 64 Head, 900K Iteration, 1024 Batch Size**
6	**0.39**	0.288	16 Block, 16 Head, 900K Iteration, 1024 Batch Size

On the contrary, we present some useless efforts.

1. We construct different training sets by generating samples using various trajectory lengths, such as 100 trajectories, 200 trajectories, and 300 trajectories. Then we combine them as several training sets to learn skills. As shown in row 4 of Table 2, we achieve improvement on the training set but not on testing set.
2. We also tried some different transformer variants, such as Bhojanapalli et al. (2020). Regrettably, as shown in row 6 of Table 4, no improvement is achieved.
3. We believe that approaches that perform poorly on the training set will also not achieve good results on the testing set. Hence, we did not submit some solutions to the challenge since they do not get promising results on the training set. This kind of solution includes replacing 64 heads with 256 heads, replacing 12 blocks with 18 blocks, and different learning rates.

Table 4: The submission result on Pushing Chair task

Row	Training Success Rate	Testing Success Rate	Detail
1	0.18	0.130	6 Block, 4 Head, 150K Iteration, 128 Batch Size(baseline)
2	0.26	**0.232**	**12 Block, 16 Head, 900K Iteration, 1024 Batch Size**
3	0.30	0.224	6 Block, 64 Head, 900K Iteration, 1024 Batch Size
4	**0.32**	**0.232**	**12 Block, 64 Head, 900K Iteration, 1024 Batch Size**
5	0.26	0.164	16 Block, 16 Head, 900K Iteration, 1024 Batch Size
6	0.31	0.224	Adopt a transformer variant Bhojanapalli et al. (2020)

Table 5: The final result and ranking of all participating teams

Rank	Team Name	Final Score	Moving Bucket	Opening Door	Opening Drawer	Pushing Chair
1	Silver-Bullet-3D	0.5740	0.602	0.552	0.744	0.398
2	Fattonny	0.4070	0.328	0.462	0.512	0.326
3	bigfish(ours)	0.3435	0.310	0.394	0.466	0.204
4	MI	0.3320	0.212	0.412	0.454	0.25
5	SieRra11799	0.1890	0.170	0.122	0.300	0.166
6	ic	0.1880	0.084	0.176	0.356	0.136
7	zhihao	0.1835	0.140	0.170	0.268	0.156

Table 5 shows the final result and ranking of all participating teams. Finally, we got a top-3 ranking in this no interaction track of the ManiSkill challenge. More importantly, to promote the research on manipulating unseen objects in the SAPIEN simulator, we release our models and scripts on Github as a new baseline.

5 Conclusion

In this technical report, we present a brief description of our solution to the no interaction track of SAPIEN ManiSkill Challenge 2021. Our method utilizes an end-to-end pipeline which mainly consists of two steps: first, we extract the point cloud features of multiple objects; then we adopt these features to predict the action score of the robot simulators via a transformer-based network. More specially, to give guidance for future work, we provide an empirical study that includes a bag of tricks and some useless attempts. Finally, our method achieves a top ranking on the leaderboard. In future works, we will further explore to manipulate unseen objects through designing suitable learning from demonstrations (LfD) algorithms.

6 Acknowledgement

This work is supported in part by National Key RD Program of China (2019YFB2102202).

References

Srinadh Bhojanapalli, Chulhee Yun, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Low-rank bottleneck in multi-head attention models. In *International Conference on Machine Learning*, pp. 864–873, 2020.

R Charles, Y Li, S Hao, and J Pointnet+ Leonidas. Deep hierarchical feature learning on point sets in a metric space. *Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA*, pp. 4–9, 2017.

Jiafei Duan, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan. A survey of embodied ai: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022.

Yueqi Duan, Yu Zheng, Jiwen Lu, Jie Zhou, and Qi Tian. Structural relational reasoning of point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 949–958, 2019.

Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(12):4338–4364, 2020.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Fei Xia, William B Shen, Chengshu Li, Priya Kasimbeg, Micael Edmond Tchapmi, Alexander Toshev, Roberto Martín-Martín, and Silvio Savarese. Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 5(2): 713–720, 2020.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11097–11107, 2020.