SILVER-BULLET-3D AT MANISKILL 2021: LEARNING-FROM-DEMONSTRATIONS AND HEURISTIC RULE-BASED METHODS FOR OBJECT MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper presents an overview and comparative analysis of our systems designed for the following two tracks in SAPIEN ManiSkill Challenge 2021 [1]:

No Interaction Track: The No Interaction track targets for learning policies from pre-collected demonstration trajectories. We investigate both imitation learning-based approach, *i.e.*, imitating the observed behavior using classical supervised learning techniques, and offline reinforcement learning-based approaches, for this track. Moreover, the geometry and texture structures of objects and robotic arms are exploited via Transformer-based networks to facilitate imitation learning.

No Restriction Track: In this track, we design a Heuristic Rule-based Method (HRM) to trigger high-quality object manipulation by decomposing the task into a series of sub-tasks. For each sub-task, the simple rule-based controlling strategies are adopted to predict actions that can be applied to robotic arms.

To ease the implementations of our systems, all the source codes and pre-trained models are available at `https://github.com/caiqi/Silver-Bullet-3D/`.

1 INTRODUCTION

It is not trivial to learn generalized policies for object manipulation from 3D visual inputs. The difficulty arises from the complexity of controlling robotic arms of multi-degree-of-freedom (multi-DOF) and the diversity of 3D shapes. Most previous benchmarks are insufficient for learning generalizable manipulation skills. On the one hand, many existing manipulation tasks Kolve et al. (2017); Wu et al. (2018); Puig et al. (2018); Xia et al. (2018; 2020) only support high-level planning with abstract action space and are thus unsuitable for studying more challenging scenarios which require low-level full-physics simulation. On the other hand, most benchmarks Zhu et al. (2020); Urakami et al. (2019); Yu et al. (2020); James et al. (2020); Dosovitskiy et al. (2017); Xu et al. (2019) lack either task-level diversity or object-level variances. Compared with previous benchmarks, ManiSkill Mu et al. (2021) is much more challenging but better resembles common real-world robotics setups. In this work, we aim at exploiting imitation learning and offline reinforcement learning to learn generalizable manipulation skills from pre-collected demonstrations. In addition, we also explore the possibility of interacting with the simulation environments and design a Heuristic Rule-based Method (HRM).

2 LEARNING-FROM-DEMONSTRATIONS IN NO INTERACTION TRACK

2.1 SYSTEM

For No Interaction Track, we design our system in two directions: imitation learning and offline reinforcement learning. In imitation learning, the policy is trained with supervised learning that directly imitates the behaviors of provided demonstrations. More specifically, give the observation (s) - action (a) pairs as training samples, the imitation learning optimizes the function by approximating $f_\theta(s) \rightarrow a$. Unlike imitation learning which highly hinges on the quality of the data-collecting

[1] `https://sapien.ucsd.edu/challenges/maniskill2021/`

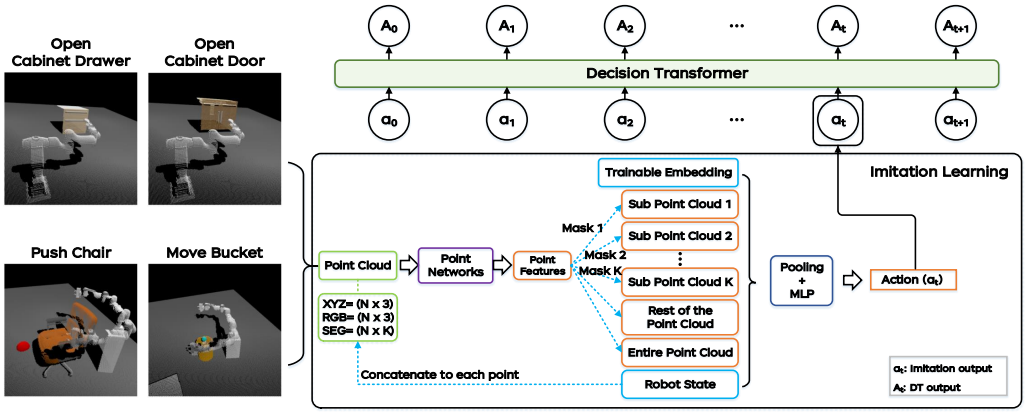

Figure 1: An overview of our solution for No Interaction Track by exploiting imitation learning and Decision Transformer.

process, offline reinforcement learning is more tolerant to the noise of demonstrations. Therefore, we take inspirations from the Decision Transformer (DT) Chen et al. (2021) and cast the task of manipulation objects as a sequence modeling problem. Figure 1 illustrates the overview of our Imitation Learning and Decision Transformer architecture (namely IL-DT), which will be described in the following sections.

2.1.1 Imitation Learning

Each demonstration consists of a sequence of observations $s = (p, c, m, g)$ and actions a, where p is the observed point cloud and c is the color of point cloud. m denotes the mask for point cloud, which groups the point cloud into several parts like robotic arms, cabinet doors, etc. g represents the state of robotic arms. To model the geometric information of objects and robotic agents, we calculate the relative distance between different parts, which is regarded as geometric feature. More specifically, by denoting the finger position and center position of the robotic agent as g_f and g_c respectively, the geometric feature is measured as:

$$x = concat(p, p - g_f, p - g_c). \tag{1}$$

Next, a three-stage network is designed to learn representative features to manipulate objects.

Point-level features. The feature of each point is composed by two parts: geometric information and texture information. Specifically, the geometric information defined in Eq. (1) is fed into an MLP network and the RGB information c is fed into another MLP network to extract features. The state of robotic arms g is further embedded into dense robotic features. The geometric features, texture features, and robotic features are concatenated as the point-level representation for each point.

Part-level features. Since different parts of the manipulated objects and robotic arms contain diverse topological and geometric properties, we then group the points according to the provided mask m. Each group of points is processed by a separate MLP network to learn each part's unique features. After that, we perform mean pooling over the features of points belonging to the same part, yielding the part-level features.

Task-level features. To explore the relationships among different parts, we further employ a Transformer Vaswani et al. (2017) network over the part-level features. Specifically, the features of different parts are concatenated as a sequence, which are augmented with the self-attention mechanisms via Transformer. Finally, by concatenating the features of all parts, we feed them into a MLP network for action prediction.

To optimize the network, we combine the L1 loss and L2 loss between prediction $\hat{a}$ and target a as the objective:

$$l = ||\hat{a} - a||_2 + \beta||\hat{a} - a||, \tag{2}$$

where β denotes the loss weight that controls each component's contribution.

2.1.2 DECISION TRANSFORMER

Vallina imitation learning commonly treats each observation pair independently, leaving the temporal information within the sequence of actions unexploited. Therefore, we further design a Transformer-based network to explore the dependency over timestamps. Specifically, inspired by Decision Transformer (DT) Chen et al. (2021), we take the predictions $(\hat{a}_i, \hat{a}_2, ..., \hat{a}_T)$ from imitation learning network as the sequence input, where T is the maximum timestamp. A Transformer network absorbs the sequence and predicts refined action for each timestamp. Following Chen et al. (2021), we replace the summation/softmax over all tokens with only the previous tokens in the sequence to enable auto-regressive prediction. We leverage the pre-trained imitation learning networks to initialize the weights of DT-based networks. The overall loss is the same as Eq. (2) in imitation learning.

2.1.3 ENSEMBLE MODELS

To improve the robustness of predicted actions, we ensemble multiple models including Imitation Learning-based models and Decision Transformer-based models. We use a late fusion strategy where the actions of all models are averaged as the final prediction. Experimental results demonstrate that the ensemble models exhibit better performances than single models (see Sec. 2.2).

2.2 EXPERIMENTAL RESULTS

In this section, we compare the performances of Imitation Learning-based model, Decision Transformer-based model, and Ensemble Models for four tasks in the No Interaction track. Table 1 details the performance comparisons in between. In general, the results across different tasks consistently validate the complementarity between Imitation Learning-based model and Decision Transformer-based model.

Table 1: The performances of four manipulation tasks on No Interaction track.

Method	OpenCabinetDoor	OpenCabinetDrawer	PushChair	MoveBucket
Imitation-Learning	0.880	0.890	0.397	0.663
Decision-Transformer	0.893	0.893	0.413	0.670
Ensemble Models	0.900	0.917	0.433	0.687

Table 2 further illustrates the final performances of our submission on the leaderboard for four different manipulation tasks. The best single model achieves an average score of 0.533, which has achieved better performances than other teams. Next, by consolidating the ensembling of Imitaion Learning and Decision Transformer-based models, our ensemble system leads to 3.4% absolute improvement against the single run on the first stage evaluation. On the second stage evaluation, our ensemble model achieves the best performance (57.4%), surpassing the second team by 16.7% absolute improvement.

Table 2: Comparisons of different teams for No Interaction task on leaderboard. Results are directly taken from `https://sapien.ucsd.edu/challenges/maniskill2021/result`.

Evaluation Stage	Team	MoveBucket Train	OpenDoor Train	OpenDrawer Train	PushChair Train	MoveBucket Test	OpenDoor Test	OpenDrawer Test	PushChair Test	Final Score
First	Silver-Bullet-3D (Single)	0.632	0.920	0.968	0.408	0.488	0.176	0.280	0.392	0.533
	Silver-Bullet-3D (Ensemble)	0.744	0.944	0.968	0.456	0.528	0.168	0.400	0.328	0.567
Second	**Silver-Bullet-3D**	**0.716**	**0.896**	**0.932**	**0.468**	**0.488**	**0.208**	**0.556**	**0.328**	**0.574**
	Fattonny	0.320	0.764	0.840	0.400	0.336	0.160	0.184	0.252	0.407
	bigfish	0.360	0.664	0.740	0.256	0.260	0.124	0.192	0.152	0.344
	MI	0.268	0.700	0.788	0.320	0.156	0.124	0.120	0.180	0.332
	SieRra11799	0.196	0.200	0.456	0.212	0.144	0.044	0.144	0.120	0.190
	ic	0.108	0.280	0.484	0.156	0.060	0.072	0.228	0.116	0.188
	Zhihao	0.176	0.272	0.416	0.216	0.104	0.068	0.120	0.096	0.184

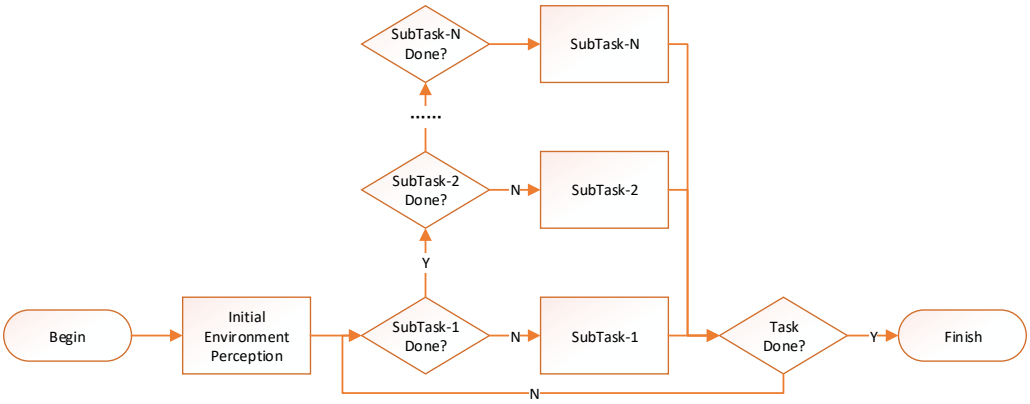

Figure 2: An overview of our Heuristic Rule-based Method (HRM).

3 Heuristic Rule-based Method in No Restriction Track

3.1 System

The basic idea of our Heuristic Rule-based Method (HRM) is to facilitate object manipulation by decomposing the task into a series of sub-tasks. Here each sub-task could be accomplished with simple rule-based controlling strategies. Figure 2 illustrates an overview of our HRM. At the beginning of a manipulation task, HRM first estimates the attributes (e.g. location, shape, size, and orientation) of the robot, target object and target point (for MoveBucket and PushChair) from the observations. This initialization process is denoted as "Environment Observation" in Figure 2. Next, in each "observation-action" iteration, the first unfinished sub-task is carried out for one step. Such iterations will stop when the robot successfully manipulates the object or takes more than 200 steps.

Based on the workflow described in Figure 2, the central core in HRM is the design of sub-tasks for each type of manipulation. Each sub-task mainly consists of two components: observation and action. Here we introduce the detailed algorithms adopted in several kinds of sub-tasks.

Sub-Task: MoveSteps (Algorithm 1). This is a simple but widely utilized sub-task. In this sub-task, the robot is designed to take the same action in a pre-determined number of steps.

Algorithm 1 Sub-Task: MoveSteps.

Input: Fixed Action Vector A, number of steps N.
Output: Action Vector α
Initialization: $i = 0$, done=False
while not done **do**
    $\alpha = A$
    Update: $i = i + 1$, done=$i \geq N$
end while

Sub-Task: MoveTo (Algorithm 2). This is another basic sub-task. Here the robot is enforced to take the same action until reaches the target location. To avoid the disturbance between different moving directions, only one direction in the action vector is activated in one sub-task. During the moving, the location of the robot could be obtained from the *state* in observation, and the target location is determined according to the object or target point. The error threshold ensures that the sub-task could be finished in a limited number of steps.

Arm Stabilization. The pose of the arms is heavily influenced by the reaction force from objects. To keep the pose of arms stable during manipulation, we design a degenerative stabilizer for arm stabilization. The stabilizer is initialized with the reference pose of the arm. Then a "MoveTo"

Algorithm 2 Sub-Task: MoveTo.

Input: Index of activated direction in action vector i, target location x_t, observation obs, velocity v, error threshold t.
Output: Action Vector α
Initialization: Basic action vector $A_0 = [0, 0, \cdots, 0]$, done=False
while not done **do**
  $x = process_obs(obs)$ ▷ Get current location from observation.
  $d = x_t - x$ ▷ Calculate distance between target and current location.
  $\alpha = A_0$
  if $d > 0$ **then**
    $\alpha[i] = v$
  else
    $\alpha[i] = -v$
  end if
  Update: done=$|d| < t$ ▷ Keep moving until the distance is smaller than the threshold.
end while

sub-task whose target is the reference pose is associated with the arm. The error threshold is set to 0.01. The output of the stabilizer is added to the action vector produced by main-stream sub-tasks.

Solutions. Based on such basic designs, we can represent the solution of the manipulation tasks as ordered lists of sub-tasks. Here we summarize the solutions for *OpenCabinetDoor*, *OpenCabinetDrawer*, *MoveBucket*, and *PushChair*.

OpenCabinetDoor

- Initialize the pose of arm (MoveSteps subtask);
- Rotate the moving platform of the robot till the robot faces the cabinet (MoveTo subtask);
- Adjust the height of the moving platform till the fingers have the same height as the handle of the door (MoveTo sub-task);
- Adjust the location along Y-axis of the moving platform till the fingers have the same Y as the handle of the door (MoveTo sub-task);
- Adjust the location along X-axis of the moving platform till the fingers have the same X as the handle of the door (MoveTo sub-task);
- Grasp the handle of the door (MoveSteps sub-task);
- Open the door (MoveSteps sub-task).

OpenCabinetDrawer

- Initialize the pose of arm (MoveSteps sub-task);
- Rotate the moving platform of the robot till the robot faces the cabinet (MoveTo sub-task);
- Adjust the height of the moving platform till the fingers have the same height as the handle of the drawer (MoveTo sub-task);
- Adjust the location along Y-axis of the moving platform till the fingers have the same Y as the handle of the drawer (MoveTo sub-task);
- Adjust the location along X-axis of the moving platform till the fingers have the same X as the handle of the drawer (MoveTo sub-task);
- Grasp the handle of the drawer (MoveSteps sub-task);
- Open the drawer (MoveSteps sub-task).

MoveBucket

- Initialize the pose of arms (MoveSteps sub-task);
- Rotate the moving platform of the robot till the robot faces the bucket (MoveTo sub-task);

- Hold the bucket with two arms (MoveSteps sub-task);
- Initialize the degenerative stabilizer for arm stabilization;
- Lift up the bucket with two arms (MoveSteps sub-task);
- Rotate the moving platform of the robot till the robot faces the target platform (MoveTo sub-task);
- Adjust the location along X- and Y-axis of the moving platform till the robot reaches the edge of the target platform (MoveTo sub-task);
- Put down the bucket (MoveSteps sub-task).

PushChair

- Move forward along X-axis for several steps to get closed to the chair (MoveSteps sub-task);
- Adjust the height of the moving platform till the arms have the same height as the armrest of the chair (MoveTo sub-task);
- Hold the chair with two arms (MoveSteps sub-task);
- Initialize the degenerative stabilizer for arm stabilization;
- Rotate the moving platform till the robot faces the target red point (MoveTo sub-task);
- Adjust the location along X- and Y-axis of the moving platform till the chair reaches the target red point (MoveTo sub-task).

3.2 Experimental Results

Table 3 illustrates the final performances of our submission on the leaderboard in No Restriction track. By consolidating the divide-and-conquer strategy which decomposes the complex task into several consecutive simple sub-tasks, we achieve a 0.928 final score, which is 23.8% absolute higher than second place. Moreover, the performances on the train set and test set are greatly reduced due to the non-data-dependent characteristic of our designed method.

Table 3: Comparisons of different teams for No Restriction task on leaderboard. Results are directly taken from `https://sapien.ucsd.edu/challenges/maniskill2021/result`.

Team	MoveBucket Train	OpenDoor Train	OpenDrawer Train	PushChair Train	MoveBucket Test	OpenDoor Test	OpenDrawer Test	PushChair Test	Final Score
Silver-Bullet-3D	**0.964**	**0.960**	**0.988**	**0.884**	**0.944**	**0.744**	**1.000**	**0.940**	**0.928**
aidan-curtis	0.772	0.472	0.972	0.732	0.572	0.280	0.988	0.728	0.690
Fattonny	0.348	0.696	0.864	0.388	0.316	0.216	0.200	0.252	0.410

4 Conclusion

In No Interaction Track of SAPIEN ManiSkill Challenge 2021, we integrate imitation learning with offline reinforcement learning, leading to high-quality learn-from-demonstrations solutions. Moreover, a three-stage network is designed to better exploit geometric and color information of observations. For No Restriction Track, we design a Heuristic Rule-based Method for object manipulation. Our future works include more in-depth studies of how imitation learning and offline reinforcement learning could be fused to boost policy learning and how to improve the performances with deep online reinforcement learning.

References

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34, 2021.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pp. 1–16. PMLR, 2017.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 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 Cathera Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao 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.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8494–8502, 2018.

Yusuke Urakami, Alec Hodgkinson, Casey Carlin, Randall Leu, Luca Rigazio, and Pieter Abbeel. Doorgym: A scalable door opening environment and baseline agent. *arXiv preprint arXiv:1908.01887*, 2019.

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.

Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*, 2018.

Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9068–9079, 2018.

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.

Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. Deep view synthesis from sparse photometric images. *ACM Transactions on Graphics (ToG)*, 38(4):1–13, 2019.

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*, pp. 1094–1100. PMLR, 2020.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.