# Frame Mining: a Free Lunch for Learning Robotic Manipulation from 3D Point Clouds

**Minghua Liu[1*], Xuanlin Li[1*], Zhan Ling[1*], Yangyan Li[2], Hao Su[1]**
[1]UC San Diego [2]Alibaba

**Abstract:** We study how choices of input point cloud coordinate frames affect learning of manipulation skills from 3D point clouds. There exist a variety of coordinate frame choices to normalize captured robot-object-interaction point clouds. We find that different frames have a profound effect on agent learning performance, and the trend is similar across 3D backbone networks. In particular, the end-effector frame and the target-part frame achieve higher training efficiency than the commonly used world frame and robot-base frame in many tasks, intuitively because they provide helpful alignments among point clouds across time steps and thus can simplify visual module learning. Moreover, the well-performing frames vary across tasks, and some tasks may benefit from multiple frame candidates. We thus propose FrameMiners to adaptively select candidate frames and fuse their merits in a task-agnostic manner. Experimentally, FrameMiners achieves on-par or significantly higher performance than the best single-frame version on five fully physical manipulation tasks adapted from ManiSkill and OCRTOC. Without changing existing camera placements or adding extra cameras, point cloud frame mining can serve as a free lunch to improve 3D manipulation learning.

**Keywords:** point cloud, coordinate frame, robot manipulation, 3D, RL

## 1 Introduction

With the rapid development and proliferation of low-cost 3D sensors, point clouds have become more accessible and affordable in robotics tasks [1]. Also, the tremendous progress in building neural networks with 3D point clouds [2, 3, 4, 5, 6, 7] has enabled powerful and flexible frameworks for 3D visual understanding tasks such as 3D object detection [5, 8, 9], 6D pose estimation [10, 11], and instance segmentation [12, 13]. Very recently, point cloud started to be used as the input to deep reinforcement learning (RL) for object manipulation [14, 15, 16], which aims at learning mappings directly from raw 3D sensor observations of unstructured environments to robot action commands. These end-to-end learning methods avoid highly structured pipelines and laborious human engineering required by conventional robot manipulation systems.

When building an agent with point cloud input, existing works [14, 15, 16] typically incorporate off-the-shelf point cloud backbone networks (e.g., PointNet [2]) into the pipeline as a feature extractor of the scene. However, some facets in constructing point cloud representations have been overlooked. For example, in the literature of 3D deep learning, the choice of coordinate frame significantly affects task performance [2, 17, 18, 19, 20, 21]. On 3D instance segmentation benchmarks for autonomous driving, previous work such as [5] showed a pipeline to process input point clouds in the camera frame, frustum frame, and object frame subsequently, leading to a large performance boost in comparison to using the camera frame alone. For our goal of manipulation skill learning, point clouds describe dynamic interactions between robots and objects, including frequent contacts and occlusions. This is a novel and more complex setting that differs from well-explored scenarios in 3D supervised learning (e.g., single objects, outdoor scenes for autonomous driving). Under this setting, choices of coordinate frames are more flexible and diverse as multiple entities (e.g., robot and manipulated object) and dynamic movements are involved.

*equal contribution

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

In this work, we first examine whether and how different coordinate frames may affect the performance and sample efficiency of point cloud-based RL for object manipulation tasks. We study four candidate coordinate frames: world frame, robot-base frame, end-effector frame, and target-part frame. These frames differ in positions of origin and orientations of axes, and canonicalize inputs in different manners (e.g., a fixed third-view, ego-centric, hand-centric, object-centric). The comparison and analysis are performed on five distinct physical manipulation tasks adapted from ManiSkill [22] and OCRTOC [23], covering various numbers of arms, robot mobilities, and camera settings. Results show that the choice of frames has profound effects. In particular, the end-effector frame and the target-part frames, rarely considered in previous works, lead to significantly better sample efficiency and final convergence than the widely used world frame and robot-base frame on many tasks. Visualization and analysis indicate that, by using different coordinate frames to represent input point clouds, we are actually performing various alignments of input scenes through $\mathbb{SE}(3)$ transformations, which may simplify the learning of visual modules.

However, the well-performing single coordinate frame may vary from task to task, and in many cases, we may need coordination between decisions made according to multiple coordinate frames. For example, tasks equipped with dual-arm robots may benefit from both left-hand and right-hand frames. For mobile manipulation tasks involving both navigation and manipulation, different frames could favor different skills (e.g., robot-base frame for navigation skills, end-effector frame for manipulation skills). We thus propose three task-agnostic strategies to adaptively select from multiple candidate coordinate frames and fuse their merits, leading to more efficient and effective object manipulation policy learning. Because we do not need to capture additional camera views or rely on task-specific frame selections, our frame mining strategies can be used as a free lunch to improve existing methods on point cloud-based policy learning. We call these fusion approaches as *FrameMiners*. Experimentally, we find that it matters to fuse information from multiple frames, but the specific FrameMiner to choose does not create much performance difference. In particular, we use one of the FrameMiners, MixAction, to interpret the importance of different frames in the policy execution process, and the interpretation agrees with our intuitions.

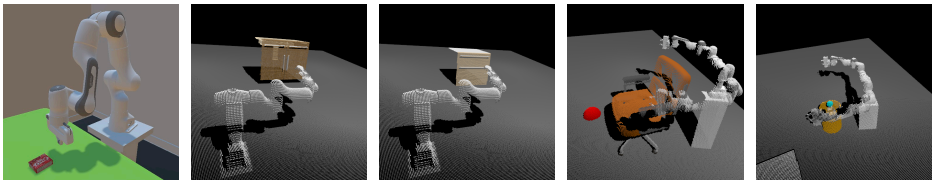
In summary, the main contributions of this work are as follows:

- We find that the choice of coordinate frame has a profound impact on point cloud-based RL for object manipulation. In particular, the end-effector frame and the target-part frame lead to much better sample efficiency than the widely-used world frame and robot-base frame on many tasks;
- We find that well-performing frames differ task by task, necessitating task-agnostic ways to select and fuse frames. This observation is consistent across 3D backbone networks;
- We propose FrameMiners, a collection of methods to fuse information from multiple candiate frames. FrameMiners provide a free lunch to improve existing point cloud-based manipulation learning methods without changing camera placements or requiring additional camera views.

## 2 Related Work

**Manipulation Learning with Point Clouds** Visual representation learning for object manipulation has been extensively studied [24, 25, 26, 27, 28, 29, 30, 31]. With the flourishment of 3D deep learning [2, 3, 4, 5, 6, 7], a major line of work learns representations from 3D point clouds for object manipulation [32, 33, 34, 35, 36, 37, 38]. Recently, people have also started to incorporate point clouds into deep reinforcement learning (RL) pipelines for manipulation learning [14, 15, 16]. However, existing point cloud-based manipulation learning methods have not paid enough attention to coordinate frame selections of input point clouds, which is fundamental in 3D visual learning. Some very recent work [39, 40] explored placement and selection of camera views and fusion of multi-view images. We differ from them in that we focus on the preprocessing of captured input point clouds without modifying existing camera configurations or adding additional cameras.

**Normalization and View Fusion in Point Cloud Learning** Normalizing input point clouds is a common practice in 3D deep learning literature. For example, in single object analysis (e.g., classification and part segmentation), people often normalize input point clouds into a categorical canonical pose with unit scale [2, 3], simplifying network training. Prior works find that existing point cloud networks [2, 3, 7, 41, 42] are very sensitive to input normalization [21, 43, 44], and



| Task | PickObject | OpenCabinetDoor | OpenCabinetDrawer | PushChair | MoveBucket |
|---|---|---|---|---|---|
| Robot Arm | Single Arm | Single Arm | Single Arm | Dual Arm | Dual Arm |
| Mobility | Fixed Base | Mobile | Mobile | Mobile | Mobile |
| Cameras | 3rd-View | Panoramic on Head | Panoramic on Head | Panoramic on Head | Panoramic on Head |

Figure 1: We study coordinate frame mining on manipulation tasks adapted from OCRTOC [23] and ManiSkill [22] covering various setups (e.g., #arms, mobility, camera). Simulation is fully physical.

many recent attempts explore rotation invariant [45, 46, 47] and equivariant methods [21, 48, 49] for 3D deep learning. Compared to well-studied scenarios (e.g., single object and autonomous driving), normalizations of point clouds under robot-object interactions are under-explored.

In LiDAR point cloud learning for autonomous driving, many work focuses on the fusion of multiple views [17, 18, 19, 20]. Unlike fusing multiple camera scans, there is only one point cloud. They propose to process the point cloud from different views (e.g., perspective view and birds-eye view) to combine their merits, which has proven to be helpful. Our work shares a similar idea, but we focus on robotic object manipulation settings, and the choice of coordinate systems is more diverse.

# 3 Point Cloud Coordinate Frame Selection Matters

## 3.1 Problem Setup

We aim to learn agents with point cloud input for object manipulation tasks via RL. A task is formally defined as a Partially-Observable Markov Decision Process (POMDP), which is represented by a tuple $M = (S, A, \mu, T, R, \gamma, \Omega, O)$. Here $S$ and $A$ are the environment state space and the action space. $\mu(s)$, $T(s'|s,a)$, $R(s,a)$, and $\gamma$ are the initial state distribution, state transition probability, reward function, and discount factor, respectively. $O(s) : S \rightarrow \Omega$ is the observation function that maps environment states to the observation space $\Omega$. Our agent is represented by a policy $\pi : \Omega \rightarrow A$, which aims to maximize the expected accumulated return given by $J(\pi \circ O) = E_{\mu,T,\pi}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$. Note that $\pi$ does not have access to the environment state $s$ and only has access to the observation $O(s)$. In this work, $O(s)$ consists of two parts: (1) a 3D point cloud captured by depth cameras; (2) proprioceptive states for the robot, such as joint positions and joint velocities. For the first part, if there are multiple cameras, we fuse all point clouds into a single one by transforming them into the same coordinate frame and concatenating the points together.

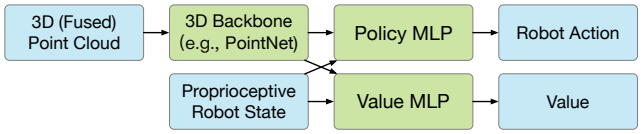



Figure 2: Architecture of a 3D point cloud-based agent, which is optimized by actor-critic RL algorithms. We study coordinate frame selection of input (fused) point cloud.

Fig. 2 shows the architecture of a 3D point cloud-based agent, which we use to discuss in this section. It first exploits a 3D backbone (e.g., PointNet [2]) to extract visual features from a 3D (fused) point cloud. The extracted features are then concatenated with proprioceptive robot states and fed into separate multi-layer perceptrons (MLP) for action and value prediction. The input (fused) point cloud can be represented in different coordinate frames before being fed into the 3D backbone network, and *the choice of coordinate frame is independent of camera views*. For example, a point cloud captured by a camera mounted on the robot's head can be transformed into the end-effector frame. In this work, we study how point cloud coordinate frames affect sample efficiency and final convergence of object manipulation learning. Unlike prior works [39, 40], we do not change robot camera configurations (e.g., camera placement, inclusion of additional cameras).

As shown in Fig. 1, we exemplify the frame selection problem on five fully-physical manipulation tasks, covering various numbers of robot arms, mobilities, and camera settings. Among them, Pick-

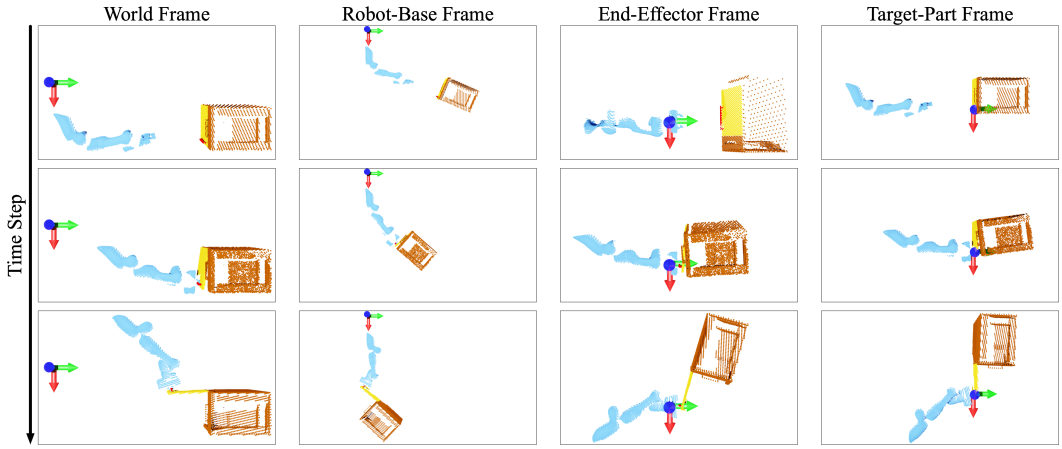


Figure 3: Illustration of four coordinate frames, which provide different alignments across time steps. We visualize three point clouds (three time steps) of an OpenCabinetDoor trajectory. Each row shows the same point cloud represented in different coordinate frames. Please zoom in for details. Robot arm, cabinet door handle, cabinet door, and cabinet body are colored in blue, red, yellow, and brown, respectively. RGB arrows indicate the corresponding origin and axes for each frame. Since the point clouds used for policy learning can be rather sparse, we show dense point clouds here for better visualization.

Object is adapted from OCRTOC [23], and the other four tasks are adapted from ManiSkill [22]. On PickObject, a fixed-base single-arm robot learns to physically grasp an object from the table, lift it up to a target height, and keep it static for a while. Point clouds are captured from a 3rd-view camera. On ManiSkill tasks, agents learn generalizable physical manipulation skills (i.e., opening cabinet doors / drawers, pushing chairs / moving buckets to target positions) across objects with diverse topology, geometry, and appearance. We utilize mobile robots with one or two arms. Point clouds come from a panoramic camera mounted on the robot's head. Action space includes joint velocities of the arm(s) and the mobile robot base, along with joint positions of the gripper(s).

### 3.2 Choices of Point Cloud Coordinate Frame

For 3D supervised learning tasks such as object classification and detection, it's a common practice to normalize input point clouds, and the choice of coordinate frames significantly affects task performance [2, 21, 17, 18, 19, 20]. In point cloud-based manipulation learning, we are faced with an underexplored, yet more challenging, setting. First, point clouds describe more complex robot-object interactions, possibly including frequent contacts and occlusions. Furthermore, compared to supervised learning, 3D visual modules receive weaker supervision signals during RL training. Therefore, it may become even more important to lessen the burden of visual module learning by properly normalizing input point clouds. Unlike previous well-studied point cloud learning scenarios (e.g., single-object point clouds, LiDAR point clouds for autonomous driving), there exist more diverse choices of coordinate frames. In this paper, we compare and analyze four candidates:

- A **world frame** is attached to a fixed point in the world (e.g., the start point of a trajectory).
- A **robot-base frame** is attached to the robot base, offering an egocentric perspective on a mobile robot. For a fixed-base robot, world frame and robot-base frame could be equivalent.
- In many object manipulation tasks, movements of robot end-effector(s) play important roles, and we can attach an **end-effector frame** to each of them. Note that for dual-arm robots, there are two end-effectors and thus two end-effector frames.
- A **target-part frame** is attached to the object part the robot intends to interact with (e.g., target door handle for the OpenCabinetDoor task).

When we transform captured point clouds into the world frame, the robot-base frame, and the end-effector frame, we may need proprioceptive robot states and potential robot movement tracking, which is typically accessible in modern robots. When we transform point clouds into the target-part frame, we may need to leverage off-the-shelf 3D object detection and pose estimation techniques.

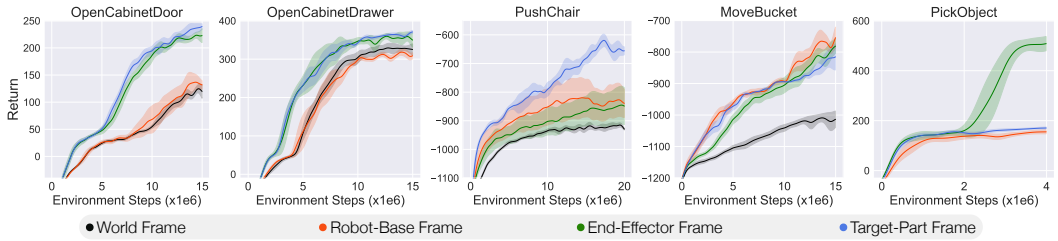



Figure 4: Comparison of four coordinate frames on five fully-physical manipulation tasks. The (fused) point cloud is transformed to a *single* coordinate frame before being fed to the visual backbone network. For dual-arm tasks (i.e., PushChair and MoveBucket), we use the right-hand frame as the end-effector frame. For PickObject, which has a fixed base, the world frame is the same as the robot-base frame. Mean and standard deviation over 5 seeds are shown.

However, in this paper, we mainly focus on the choices of coordinate frames themselves. In our simulated experiments, we use ground truth object poses for the target-part frame.

In Fig. 3, we visualize an example trajectory under four coordinate frames. As shown in the figure, different coordinate frames canonicalize inputs in different manners (e.g., a fixed third-view, egocentric, hand-centric, and object-centric), which is essentially performing various *alignments* among point clouds across multiple time steps. For example, in the end-effector frame, the end-effector is always aligned at the origin throughout a trajectory. Such alignments may simplify the learning of visual modules in distinct ways. With the end-effector frame, the network does not need to locate the end-effector in point clouds (always at the origin). Similarly, with the target-part frame, it can be easier to determine the relative position between the target part and the robot end-effector. The robot-base frame naturally aligns its frame axes with the moving directions of the robot's base.

### 3.3 Single-Frame Comparison on Manipulation Tasks

We compare the four coordinate frames on the five manipulation tasks by training PPO [50] agents using PointNet [2] as the 3D visual backbone. In this section, the (fused) point cloud is transformed into a **single** coordinate frame. For PushChair and MoveBucket tasks that use a dual-arm robot, we use the right hand frame as the end-effector frame (we observe almost identical performance using the left hand frame). For the target-part frame, we choose the handle frame for OpenCabinetDoor and OpenCabinetDrawer tasks, chair seat frame for the PushChair task, bucket for the MoveBucket task, and the target object for the PickObject task. Further details are presented in the supplementary.

Fig. 4 shows the results. We observe that distinct coordinate frames lead to very different agent training performances. Overall, the world frame is the least effective, especially in PushChair and MoveBucket that involve more pronounced movement of the robot base. This suggests that the alignment of a static point in the world-frame is less helpful for the tasks. Compared to the commonly-used world-frame and robot-base frame, the end-effector frame has much higher sample efficiency on all single-arm tasks (i.e., OpenCabinetDoor, OpenCabinetDrawer, and PickObject), demonstrating the benefits of end-effector alignment. However, it shows similar or worse performance on PushChair and MoveBucket, intuitively because these tasks rely on dual-arm coordination, but our point cloud is normalized to a single end-effector frame (i.e., right hand frame). In addition, the target-part frame achieves the best sample efficiency on most tasks, suggesting that the target-part alignment across time could be of great help for point cloud-based visual manipulation learning.

### 3.4 Further Analysis

In robot manipulation tasks, the network needs to infer binary relations between subjects (e.g., relative pose between the end-effector and the cabinet handle). By aligning point clouds under certain frames (e.g., end-effector frame), these tasks may be reduced to single-subject location tasks (e.g., simply copying the handle pose), which become much easier to solve. To confirm this hypothesis, we perform a diagnosis experiment on OpenCabinetDoor, where we intentionally remove all robot points (i.e., blue points in Fig. 3) and see its effect on different coordinate frames. As shown in Fig. 5, after the robot points are removed, the end-effector frame performs the same, while the robot-base

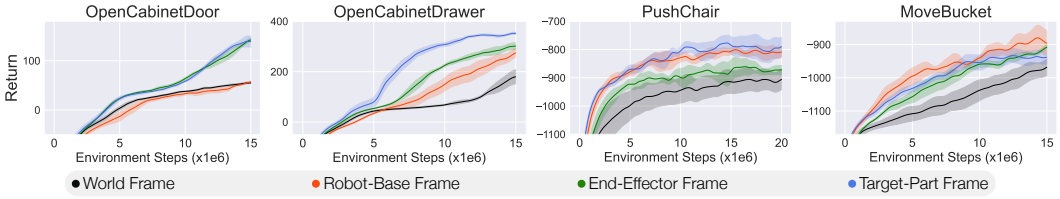



Figure 6: Using SparseConvNet [51] as the 3D visual backbone, we observe similar trends as Fig 4. Mean and standard deviation over 5 seeds are shown.

frame performs worse (the task is still solvable since the end-effector position is also provided in the proprioceptive robot state). This suggests that the end-effector frame allows an agent to focus on the target object, along with its interaction with the robot hand, which verifies our hypothesis.

We utilized PointNet [2] as our 3D visual backbone for its fast speed and general good performance. However, it's unclear whether point cloud frame selection is also crucial for other 3D backbones, especially those more complex and powerful. Therefore, we conduct the same experiments as Sec 3.3 using SparseConvNet [51], a heavier 3D backbone network, on the four ManiSkill tasks (further details in the supplementary). As shown in Fig. 6, we observe similar relative performance between frames as before (e.g., the world frame performs poorly; the end-effector frame outperforms the world frame and the robot-base frame on OpenCabinetDoor and OpenCabinetDrawer). Interestingly, using SparseConvNet doesn't improve the overall performance over PointNet.

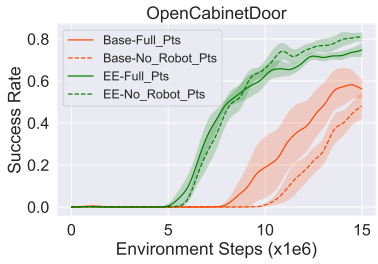



Figure 5: Removing robot points does not harm the performance of the EE frame on OpenCabinetDoor.

# 4 Mining Multiple Coordinate Frames

We have shown that different point cloud coordinate frames lead to distinct sample efficiencies and final performances in manipulation learning. However, a single frame can perform well on some tasks but poorly on others, and we wish to find good frames in a task-agnostic manner. Moreover, for complex manipulation tasks, a single frame could be insufficient, and synergistic coordination between multiple frames could provide unparalleled advantages. For example, when robots are equipped with multiple arms, each arm may have its preferred coordinate frame (e.g., left-hand frame and right-hand frame). In addition, in tasks that involve simultaneous navigation and manipulation (e.g., on PushChair and MoveBucket, an agent needs to move towards the target while manipulating chairs or buckets), different frames could benefit different skills (e.g., robot-base frame for navigation skills, and end-effector frame for manipulation skills). Therefore, it is of great help to propose a prior-agnostic method that can automatically select the best frame from multiple candidates or combine the merits of them. Again, we are not talking about fusing multiple camera views. Point clouds from multiple camera views are first fused together into a single point cloud, before being transformed to each coordinate frame.

In this section, we will present a collection of three strategies to adaptively select and fuse multiple candidate coordinate frames, and we call them *FrameMiners*. In particular, we will first introduce *FrameMiner-MixAction* in Section 4.1 in detail to interpret the importance of different frames in the policy execution process. We will then briefly introduce the other two FrameMiners and compare different approaches with single-frame baselines.

## 4.1 FrameMiner-MixAction

Inspired by the idea of mixture of experts [52], we propose a general and interpretable framework, *FrameMiner-MixAction* (FM-MA). As shown in Fig. 7, FM-MA takes a (fused) point cloud and $n$ candidate coordinate frames as input, and first transforms the point cloud into $n$ coordinate frames. For each transformed point cloud, FM-MA employs an expert network, consisting of a 3D visual backbone (e.g., PointNet [2]) and an MLP, to propose full robot actions (e.g. target velocity of $m$ joints). Since different experts use different coordinate frames, they are encouraged to specialize

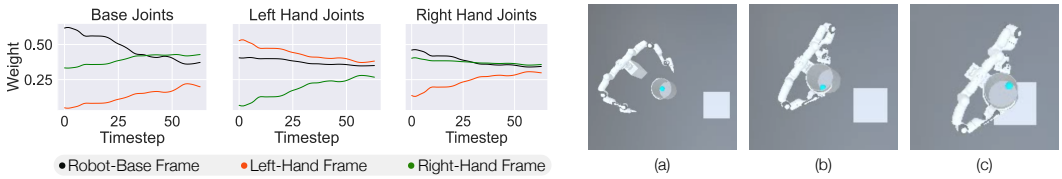



Figure 8: Left: learned weights of FrameMiner-MixAction over a MoveBucket trajectory, where three coordinate frames are fused. We divide robot joints into three groups and show the average weights of each group in each coordinate frame. Right: three stages of the trajectory. (a) Approaching the bucket. (b) Moving the bucket to the platform. (c) Placing the bucket on the platform.

different skills and controls of different joints. Finally, we combine actions from the $n$ experts through input-dependent weights. Specifically, we concatenate extracted visual features from all $n$ frames with the proprioceptive robot state, feed it into an MLP, and predict a weight for each pair of expert and joint (there are $n \times m$ weights in total). For each joint, we normalize the weights over $n$ experts via softmax and fuse the actions through weighted linear combination.

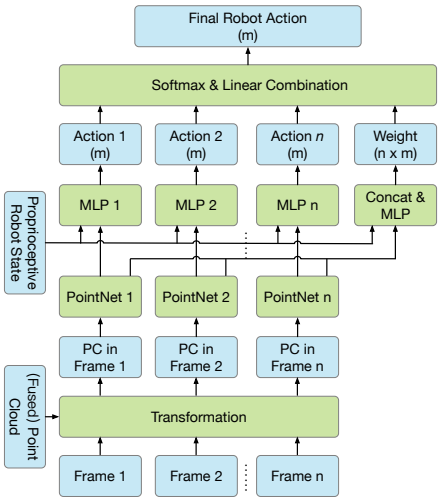



Figure 7: The pipeline of FrameMiner-MixAction (FM-MA). Each frame outputs an action proposal. Actions are then fused through input-dependent and joint-specific weights.

FM-MA fuses actions by predicting joint-specific weights, since we believe that, for different joints, we need to extract information from different coordinate frames. Furthermore, the weights are input-dependent, potentially allowing the model to capture dynamic joint-frame relations at different stages of a task. Fig. 8 confirms these hypotheses. On MoveBucket trajectories, we observe distinct frame preferences between different robot joints. The left and right-hand frames contribute significantly to their respective joint actions. In addition, the weight distribution changes greatly over different trajectory stages. Initially, when the robot is moving towards the bucket but not interacting with it, the base frame contributes more. However, when hands start to manipulate the bucket, the weights of the hand frames increase. In particular, when the robot places the bucket onto the platform, we need careful coordination among all joints, and thus similar weights from each frame.

### 4.2 Comparison of Different FrameMiners

To study how network architectures influence coordinate frame fusion, we also propose other two strategies: *FrameMiner-FeatureConcat* (FM-FC) and *FrameMiner-TransformerGroup* (FM-TG). For each transformed point cloud, FM-FC uses an individual PointNet to extract visual feature. All visual features are then concatenated and fed into an MLP to predict robot action. FM-TG decomposes our robot action into three groups: base joint actions, left-hand joint actions, and right-hand joint actions (only two groups for single-arm tasks). After visual features are extracted from PointNets, they are fused through a Transformer [53] to produce a feature for each action group, which passes through an MLP to predict its respective joint actions (see supplementary for details).

We compare our three FrameMiners with single-frame baselines. Specifically, in this section, we focus on frame mining among the robot-base frame and end-effector frame(s). For the dual-arm tasks (i.e., PushChair and MoveBucket), the end-effector frames include both the left-hand frame and the right-hand frame. We will discuss the inclusion of target-part frame in Section 4.3. Fig. 9 and Tab. 1 show the comparison results. On single-arm tasks (i.e., OpenCabinetDoor/Drawer), our FrameMiners perform on par with the end-effector frame, which suggests that FrameMiners can automatically select the best single frame.

| | Base | EE | FM-FC | FM-MA | FM-TG |
|---|---|---|---|---|---|
| Door | 54±7 | 80±2 | 79±3 | **84±2** | 70±6 |
| Drawer | 88±2 | 93±1 | **94±1** | 93±1 | 93±2 |
| Chair | 7±3 | 2±1 | 32±4 | **36±4** | 34±6 |
| Bucket | 23±6 | 19±4 | 77±5 | 81±3 | **90±2** |

Table 1: Success rates (%) on four ManiSkill tasks.

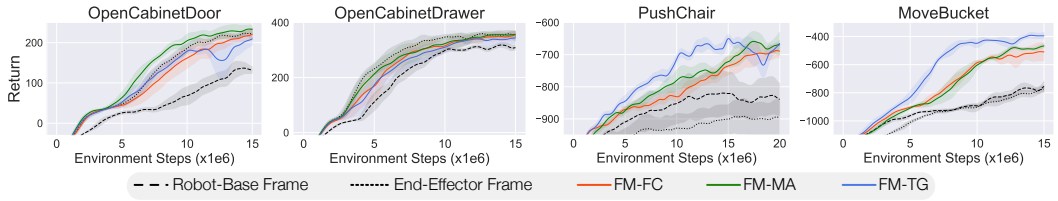



Figure 9: Comparison of different frame mining approaches on the four ManiSkill tasks, where the robot-base frame and end-effector frame(s) are fused. Black lines indicate single-frame baselines.

On dual-arm tasks, our FrameMiners significantly outperform single-frame baselines, demonstrating the advantage of coordination between multiple coordinate frames. While it matters to fuse information from multiple frames, the specific FrameMiner to choose does not create much performance difference. Empirically, we find FM-MA less sensitive to training parameters, and FM-TG more computationally expensive.

### 4.3 Target-Part Frame

In Section 4.2, we focus on the fusion of robot-base frame and end-effector frames, since the target-part frame relies on pose estimation of target objects, which requires extra efforts in real-world settings. As shown in Fig. 10, if object poses are estimated, we sometimes observe further performance boost of FrameMiners from the target-part frame. For example, on PushChair, by incorporating the target-part frame, the success rate of FM-MA increases from 36±4% to 53±3%. On PickObject, FM-MA already achieves good performance without the target-part frame (94±2%); incorporating it slightly improves the success rate to 97±1%.

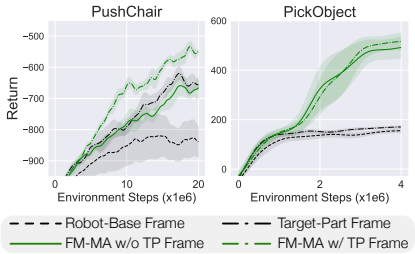



Figure 10: Fusion of target-part frame could further boost the performance.

## 5 Real World Experiments

To further verify that our learned policies can be deployed on real-world robots without introducing extra domain gaps, we test on PickObject with a Kinova Jaco2 Spherical 7-DoF robot, an Intel RealSense [54] D435 camera for uncolored point cloud capture, and a Rubik's cube from YCB objects [55, 56] (see Fig. 11). We use a 3-DoF end-effector position controller and a 1-DoF gripper position controller. We train the FM-MA policy (Section 4.1) by fusing the robot-base frame and the end-effector frame. At test time, we build a digital twin in the simulator over 25 sampled initializations of the real environment with a vision-based pipeline like Jiang et al. [57]. By following trajectories from policy rollout, we obtain a 84% success rate with FM-MA, compared to an 80% success rate with the end-effector frame. The robot-base frame is unable to achieve successful picks under our training budget. Note that the performance differences in the real world are very similar to the simulation environment, indicating that point cloud frame selection or mining does not affect original domain gap. More details are presented in supplementary.

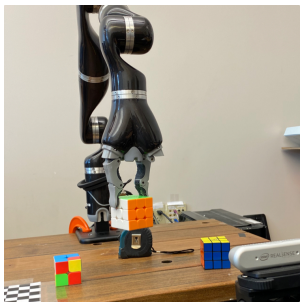


Figure 11: Real robot setup.

## 6 Conclusion and Limitations

We find that choices of point cloud coordinate frames have a profound impact on learning manipulation skills. Our proposed FrameMiners can adaptively select and fuse multiple candidate frames, serving as a free lunch for 3D point cloud-based manipulation learning. Currently, our FrameMiners need to process each frame separately, leading to more network computation. In the future, we would like to explore more advanced fusion strategies to further improve network efficiency as well as performance. In addition, the target-part frame requires human judgment to determine the target part candidates and 6D pose estimation of the target parts, although we have shown our method can also achieve great improvements without the target-part frame (Section 4.2).

**Acknowledgments**

This work is supported in part by gifts from Qualcomm. We would like to thank Jiayuan Gu and Zhiao Huang for their helpful discussion and manuscript proofreading.

## References

[1] Kinova. https://www.kinovarobotics.com, 2022. Accessed: 2022-06-14.

[2] C. R. Qi, H. Su, K. Mo, and L. 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*, pages 652–660, 2017.

[3] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[4] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[5] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018.

[6] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[7] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[8] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[9] C. R. Qi, O. Litany, K. He, and L. J. Guibas. Deep hough voting for 3d object detection in point clouds. In *proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286, 2019.

[10] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun. Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11632–11641, 2020.

[11] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3343–3352, 2019.

[12] B. Yang, J. Wang, R. Clark, Q. Hu, S. Wang, A. Markham, and N. Trigoni. Learning object bounding boxes for 3d instance segmentation on point clouds. *Advances in neural information processing systems*, 32, 2019.

[13] L. Yi, W. Zhao, H. Wang, M. Sung, and L. J. Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3947–3956, 2019.

[14] W. Huang, I. Mordatch, P. Abbeel, and D. Pathak. Generalization in dexterous manipulation via geometry-aware multi-task learning. *arXiv preprint arXiv:2111.03062*, 2021.

[15] T. Chen, J. Xu, and P. Agrawal. A system for general in-hand object re-orientation. In *Conference on Robot Learning*, pages 297–307. PMLR, 2022.

[16] Y.-H. Wu, J. Wang, and X. Wang. Learning generalizable dexterous manipulation from human grasp affordance. *arXiv preprint arXiv:2204.02320*, 2022.

[17] Y. Zhou, P. Sun, Y. Zhang, D. Anguelov, J. Gao, T. Ouyang, J. Guo, J. Ngiam, and V. Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In *Conference on Robot Learning*, pages 923–932. PMLR, 2020.

[18] Y. A. Alnaggar, M. Afifi, K. Amer, and M. ElHelw. Multi projection fusion for real-time semantic segmentation of 3d lidar point clouds. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1800–1809, 2021.

[19] M. Gerdzhev, R. Razani, E. Taghavi, and L. Bingbing. Tornado-net: multiview total variation semantic segmentation with diamond inception module. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9543–9549. IEEE, 2021.

[20] V. E. Liong, T. N. T. Nguyen, S. Widjaja, D. Sharma, and Z. J. Chong. Amvnet: Assertion-based multi-view fusion network for lidar semantic segmentation. *arXiv preprint arXiv:2012.04934*, 2020.

[21] C. Deng, O. Litany, Y. Duan, A. Poulenard, A. Tagliasacchi, and L. J. Guibas. Vector neurons: A general framework for so (3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12200–12209, 2021.

[22] T. Mu, Z. Ling, F. Xiang, D. Yang, X. Li, S. Tao, Z. Huang, Z. Jia, and H. Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021.

[23] Z. Liu, W. Liu, Y. Qin, F. Xiang, M. Gou, S. Xin, M. A. Roa, B. Calli, H. Su, Y. Sun, et al. Ocrtoc: A cloud-based competition and benchmark for robotic grasping and manipulation. *IEEE Robotics and Automation Letters*, 7(1):486–493, 2021.

[24] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.

[25] B. Chen, P. Abbeel, and D. Pathak. Unsupervised learning of visual 3d keypoints for control. In *International Conference on Machine Learning*, pages 1539–1549. PMLR, 2021.

[26] M. Vecerik, J.-B. Regli, O. Sushkov, D. Barker, R. Pevceviciute, T. Rothörl, C. Schuster, R. Hadsell, L. Agapito, and J. Scholz. S3k: Self-supervised semantic keypoints for robotic manipulation via multi-view consistency. *arXiv preprint arXiv:2009.14711*, 2020.

[27] R. Cheng, A. Agarwal, and K. Fragkiadaki. Reinforcement learning of active vision for manipulating objects under occlusions. In *Conference on Robot Learning*, pages 422–431. PMLR, 2018.

[28] Y. Zaky, G. Paruthi, B. Tripp, and J. Bergstra. Active perception and representation for robotic manipulation. *arXiv preprint arXiv:2003.06734*, 2020.

[29] R. Shah and V. Kumar. Rrl: Resnet as representation for reinforcement learning. *arXiv preprint arXiv:2107.03380*, 2021.

[30] J. Pari, N. Muhammad, S. P. Arunachalam, L. Pinto, et al. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021.

[31] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[32] X. Lin, Y. Wang, Z. Huang, and D. Held. Learning visible connectivity dynamics for cloth smoothing. In *Conference on Robot Learning*, pages 256–266. PMLR, 2021.

[33] A. Alliegro, M. Rudorfer, F. Frattin, A. Leonardis, and T. Tommasi. End-to-end learning to grasp from object point clouds. *arXiv preprint arXiv:2203.05585*, 2022.

[34] D. Yarats, I. Kostrikov, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2020.

[35] J. Lv, Q. Yu, L. Shao, W. Liu, W. Xu, and C. Lu. Sagci-system: Towards sample-efficient, generalizable, compositional, and incremental robot learning. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2022.

[36] S. James and A. J. Davison. Q-attention: Enabling efficient learning for vision-based robotic manipulation. *IEEE Robotics and Automation Letters*, 2022.

[37] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13438–13444. IEEE, 2021.

[38] B. Eisner, H. Zhang, and D. Held. Flowbot3d: Learning 3d articulation flow to manipulate articulated objects. *Robotics Science and Systems (RSS)*, 2022.

[39] K. Hsu, M. J. Kim, R. Rafailov, J. Wu, and C. Finn. Vision-based manipulators need to also see from their hands. *International Conference on Learning Representations*, 2022.

[40] R. Jangir, N. Hansen, S. Ghosal, M. Jain, and X. Wang. Look closer: Bridging egocentric and third-person views with transformers for robotic manipulation. *IEEE Robotics and Automation Letters*, 2022.

[41] M. Atzmon, H. Maron, and Y. Lipman. Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091*, 2018.

[42] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31, 2018.

[43] J. Sun, Q. Zhang, B. Kailkhura, Z. Yu, C. Xiao, and Z. M. Mao. Benchmarking robustness of 3d point cloud recognition against common corruptions. *arXiv preprint arXiv:2201.12296*, 2022.

[44] T. Lorenz, A. Ruoss, M. Balunović, G. Singh, and M. Vechev. Robustness certification for point cloud models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7608–7618, 2021.

[45] C. Chen, G. Li, R. Xu, T. Chen, M. Wang, and L. Lin. Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4994–5002, 2019.

[46] X. Li, R. Li, G. Chen, C.-W. Fu, D. Cohen-Or, and P.-A. Heng. A rotation-invariant framework for deep point cloud analysis. *IEEE Transactions on Visualization and Computer Graphics*, 2021.

[47] Z. Zhang, B.-S. Hua, D. W. Rosen, and S.-K. Yeung. Rotation invariant convolutions for 3d point clouds deep learning. In *2019 International Conference on 3D Vision (3DV)*, pages 204–213. IEEE, 2019.

[48] F. Fuchs, D. Worrall, V. Fischer, and M. Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 33:1970–1981, 2020.

[49] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[50] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[51] B. Graham and L. van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017.

[52] S. Masoudnia and R. Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2):275–293, 2014.

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[54] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik. Intel realsense stereoscopic depth cameras. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–10, 2017.

[55] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar. Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. *arXiv preprint arXiv:1502.03143*, 2015.

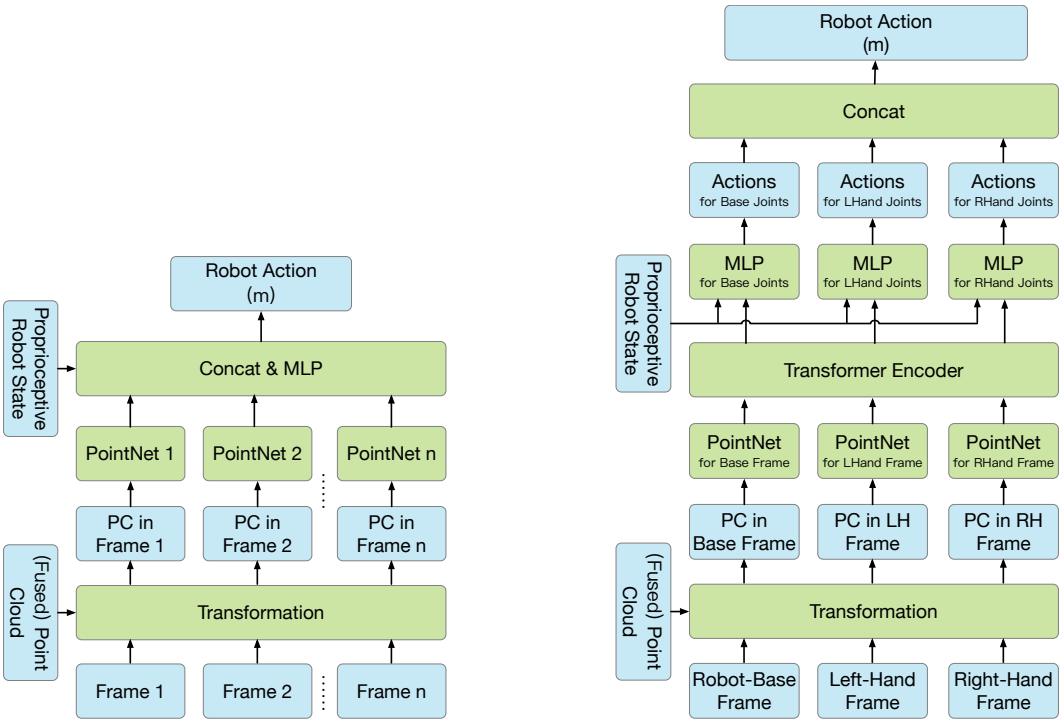
[56] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

[57] Z. Jiang, C.-C. Hsu, and Y. Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5616–5626, 2022.

[58] B. Wen, W. Lian, K. Bekris, and S. Schaal. You only demonstrate once: Category-level manipulation from single visual demonstration. In *Proceedings of Robotics: Science and Systems*, 2022.

[59] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann. Neural descriptor fields: Se(3)-equivariant object representations for manipulation. 2022.

[60] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European Conference on Computer Vision*, 2020.




(a) FrameMiner-FeatureConcat (b) FrameMiner-TransformerGroup

Figure 12: Architectures of FrameMiner-FeatureConcat and FrameMiner-TransformerGroup.

### S.1 Architecture of the other two FrameMiners

Fig. 12 shows architectures of the other two FrameMiners, FrameMiner-FeatureConcat (FM-FC) and FrameMiner-TransformerGroup (FM-TG).

### S.2 Additional Experiment Results and Discussions

#### S.2.1 Imitation Learning

In the main text, we analyzed the profound impact of coordinate frames on point cloud-based object manipulation learning through online RL algorithms. Apart from online RL, some previous work [58] have shown that dynamic selection of coordinate frames could benefit demonstration-based manipulation learning as well. In this section, we conduct experiments on imitation learning and investigate whether our previous findings can generalize to other algorithm domains.

For each task, we use an expert RL policy to generate 100 successful demonstrations. We then perform Behavior Cloning (BC) by representing input point clouds under different coordinate frames, along with using our proposed FrameMiner-MixAction (FM-MA). We utilize the same network architectures as online RL, and we use MSE loss for training. For FM-MA, the robot-base frame and the end-effector frame(s) are fused. As shown in Table 2, we observe similar findings to Section 3.3 and Section 4.2. Specifically, the end-effector frame has much higher performance on single-arm tasks (OpenCabinetDoor/Drawer), demonstrating the benefits of end-effector alignment. Our proposed FrameMiner is capable of automatically selecting the best single frame or combining the merits from multiple frames and outperforming single-frame baselines.

#### S.2.2 Alternative Designs in FM-MA (Weighted Linear Combination vs. Maximum Weight)

In the main paper, FrameMiner-MixAction (FM-MA) uses weighted linear combination to fuse action proposals from each coordinate frame (see Figure 7). For simplicity, we name this variant FM-

| | Robot-Base | End-Effector | FM-MA |
|---|---|---|---|
| OpenCabinetDoor | 50±3 | **85±3** | **83±4** |
| OpenCabinetDrawer | 72±4 | **88±2** | **88±2** |
| PushChair | 38±3 | 28±2 | **42±4** |
| MoveBucket | 76±4 | 80±2 | **91±2** |

Table 2: Behavior Cloning (BC) success rates (%) on four ManiSkill tasks. Mean and standard deviation over 5 seeds are shown.

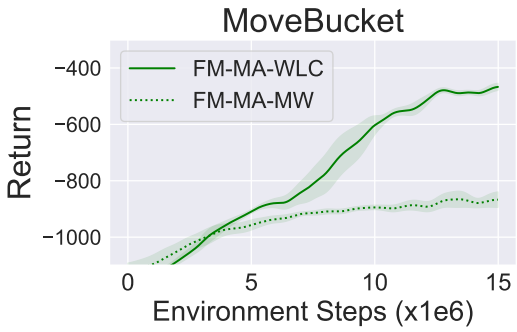



Figure 13: Comparison between FM-MA-WLC and FM-MA-MW on MoveBucket. Mean and standard deviation over 5 seeds are shown. FM-MA-WLC achieves 81±3% final success rate, while FM-MA-MW only has 9±2% final success rate.

MA-WLC. An alternative design is to choose the max-weighted action proposal for each joint (we name this variant FM-MA-MW). Formally, let $A \in \mathbb{R}^{n \times m}$, where $A_{ij}$ denotes the action proposal for the $j$-th robot joint from the $i$-th coordinate frame. Let $W \in \mathbb{R}^{n \times m}$ be the weight matrix predicted by the network. In FM-MA-MW, the output action $\mathbf{a} = (a_1, a_2, \ldots, a_m)$ satisfies $a_j = A_{kj}$ where $k = \text{argmax}_{k=1}^{n} W_{kj}$. Note that FM-MA-WLC uses SoftMax to normalize the weights; thus FM-MA-WLC can be regarded as a "soft version" of FM-MA-MW.

To compare the two designs, we conduct two experiments: (1) We train FM-MA-MW from scratch. Results are shown in Fig. 13. (2) We resume from the final checkpoint of the original FM-MA-WLC. During evaluation, we use the max-weighted action proposal as the action output. Results are shown in Table 3. We observe that for both experiments, using FM-MA-MW deteriorates performance. We conjecture that FM-MA-WLC alleviates optimization difficulty, which likely comes from the fact that it is a "soft version" of FM-MA-MW with well-behaving gradients. On the other hand, since FM-MA-MW uses argmax operation over columns of $W$, there is a lack of gradient for $W$ during training, which leads to more difficult optimization.

#### S.2.3 Ablation Study on Camera Placements

As a recap, the five tasks analyzed in our main paper cover both static and moving camera settings. The experiments in the main paper were conducted using default camera placements shown in Fig. 1. For the four tasks with moving cameras, a panoramic camera is mounted on the robot head.

While FrameMiners do not require changing existing camera placements, camera placements still matter, since different camera placements affect the point clouds being captured. Therefore, we perform an experiment where we move the panoramic camera from the robot head to the robot base. As shown in Fig. 14, we observe similar phenomena as in Fig. 9. Specifically, fusing multiple coordinate frames with our FrameMiners still leads to better sample efficiency and final performance, demonstrating that FrameMiners are robust under different camera placements.

| | FM-MA (WLC eval) | FM-MA (MW eval) |
|---|---|---|
| OpenCabinetDoor | 84±2 | 45±5 |
| OpenCabinetDrawer | 93±1 | 93±2 |
| PushChair | 36±4 | 20±3 |
| MoveBucket | 81±3 | 14±3 |

Table 3: Success rate (%) comparison between the same FM-MA checkpoint evaluated using weighted linear combination of actions (WLC) and using maximum-weighted action (MW) on four ManiSkill tasks. Mean and standard deviation over 5 seeds are shown.

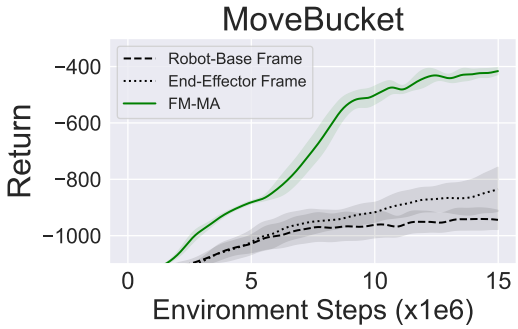



Figure 14: Results on MoveBucket with a panoramic camera mounted on the robot base. The "Robot-Base Frame" and the "End-Effector Frame" indicate the coordinate frames used to represent captured input point clouds. FM-MA fuses the two end-effector frames (left and right arms) and the robot-base frame. Mean and standard deviation over 5 seeds are shown.

#### S.2.4 Learning Adaptive Frame Transformations from Observations

In our paper, we use known transformations (e.g., end-effector pose in robot state) to align input point clouds in different coordinate frames and propose FrameMiners to fuse merits of multiple coordinate frames. A potential baseline is to learn a transformation adaptively based on input point clouds. To examine the effectiveness of this baseline, we add an additional network before the PointNet backbone to learn an adaptive $\mathbb{SE}(3)$ transformation based on the input point cloud. This transformation is then applied to the input point cloud before passing it through the PointNet backbone (note that we remove spatial transformation layers from the original PointNet in all of our experiments). However, as shown in Figure 15, adding this $\mathbb{SE}(3)$ transformation layer barely improves performance.

We conjecture that it's very difficult to predict a $\mathbb{SE}(3)$ transformation for aligning the input point cloud across time due to the large search space (where most transformations are ineffective) and weak supervision from RL training loss. Moreover, in many challenging tasks, we may need to fuse information simultaneously from multiple coordinate frames (e.g., left-hand and right-hand frames). This is not achievable through learning a single transformation. In contrast, for FrameMiners, we take advantage of easily-accessible frame information (e.g. end-effector poses) without relying on transformation prediction. We then fuse the merits of multiple candidate coordinate frames.

#### S.2.5 $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$ Equivariant Point Cloud Backbones

Recently, there have been several works on designing $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$ equivariant/invariant backbone networks for point cloud learning [21, 59]. While they are of great benefit for analysis within each object (e.g., shape classification, part segmentation, and 6D pose estimation), our robot-object interaction setting is a bit different.

In robot manipulation scenarios, a particular challenge comes from inferring the relations between two object parts (e.g., relative pose between the end-effector and the cabinet handle). This binary

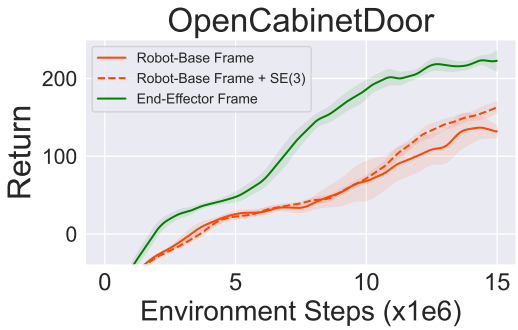



Figure 15: Ablation study on adding an adaptive $\mathbb{SE}(3)$ transformation prediction layer. When the input point cloud is represented in the robot-base frame, adding such transformation layer barely improves performance, while representing the point cloud in the end-effector frame significantly improves performance.

relation inference task is challenging under the weak RL loss supervision, even using $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$ equivariant/invariant backbones. FrameMiners explicitly approach this challenge by aligning point clouds (across multiple time steps) with the known transformation matrices (e.g., the end-effector pose). This reduces many binary relation inference tasks to single-subject location tasks, which has much lower difficulty. For example, when using the end-effector frame in the OpenCabinet task, the network only needs to copy the handle pose to infer the relative pose between the handle and the end-effector, as the end-effector is always at the frame origin.

### S.3 More Details of Manipulation Tasks

**Task Descriptions:**

- In OpenCabinetDoor, a single-arm mobile agent needs to approach a cabinet, use the handle to fully open the designated cabinet door, and then keep the door static for a while.
- In OpenCabinetDrawer, a single-arm mobile agent needs to approach a cabinet, use the handle to fully open the designated cabinet drawer, and then keep the drawer static for a while.
- In PushChair, a dual-arm mobile agent needs to approach the chair, push the chair to a target location, and then keep the chair static for a while.
- In MoveBucket, a dual-arm mobile agent needs to approach the bucket, move the bucket to a target platform, place the bucket onto the platform, and then keep the bucket static for a while.
- In PickObject, a single-arm fixed-base agent needs to grasp an object from the table, lift it up to a certain target height, and keep it static for a while.

Simulations are fully physical. For OpenCabinetDoor, OpenCabinetDrawer, PushChair, and MoveBucket, there are 66, 49, 26, and 29 different objects (designated parts) during training, respectively.

**Observations and Actions:**

For all ManiSkill tasks, the proprioceptive robot state includes:

- Positions of all (two if single-arm, four if dual-arm) fingers
- Velocities of all (two or four) fingers
- x, y position of the mobile robot base
- Mobile robot base's rotation around the z-axis
- x, y velocity of the mobile robot base
- Angular velocity of the mobile robot base around the z-axis
- Joint angles of the robot, excluding the joints in the mobile base
- Joint velocities of the robot, excluding the joints in the mobile base

- Indicator of whether each joint receives an external torque

The action space includes:

- x, y velocity of the mobile robot base
- Angular velocity of the mobile robot base around the z-axis
- Height of the robot body
- Joint velocities of the robot, excluding joints of the mobile base and the gripper fingers
- Joint positions of the gripper fingers

Joint positions of the gripper fingers are controlled by position PID. All other action components are controlled by velocity PID.

For the PickObject task, the proprioceptive robot state includes:

- Joint angles of the robot,
- Joint velocities of the robot,
- 1D gripper joint position,
- Target xyz positions of object.

The action space includes 3 DoF end-effector position and 1 DoF gripper joint position.

For all tasks, input point cloud features include xyz coordinates, RGB colors, and one-hot segmentation masks for each part category.

**Motivations for Our Task Choice**

We aim to cover a wide range of factors that may influence the selection of point cloud coordinate frames. Specifically, the tasks are chosen to cover various robot mobilities, numbers of robot arms, and camera settings, as demonstrated in Figure 1.

Different robot mobility results in differences in world frame and robot base frame. These two frames are aligned in static robots but not in mobile robots. The robot's mobility can also change the focus of tasks (e.g., navigation or object interaction), which may place different requirements on the choice of point cloud frame.

We cover both single-arm and dual-arm environments, as they pose different requirements for point cloud frame selection. In single-arm environments, using the only end-effector frame may already be able to achieve good performance. However, in dual-arm environments, there are two end-effector frames, and these tasks require precise coordination between the two robot arms, which pose significant challenges for manipulation learning. As each end-effector may have a preferred frame, the necessity of frame fusion becomes more pronounced.

Last but not least, camera placements determine sources of point clouds, which may potentially influence the selection of coordinate frames. In our experiments, we cover both static camera settings and moving camera settings (mounted on robots).

### S.4 Detailed Experimental Settings and Hyperparameters

For our visual backbones, our PointNets are implemented with a three-layer MLP with dimensions $[64, 128, 300]$ followed by a max-pooling layer. We do not apply any spatial transformation to the inputs. Our SparseConvNets are implemented as a SparseResNet10 using TorchSparse [60]. SparseResNet10 has a 4-stage pipeline with kernel size 3 and hidden channels $[64, 128, 256, 512]$ respectively. We use kernel size 3 and stride 2 for downsampling. Initial voxel size is 0.05. Final features in the final-stage voxels are maxpooled as output visual feature.

All of our agents are trained with PPO (hyperparameters in Tab. 4). Each policy MLP that outputs actions has dimensions $[192, 128, \text{action_dim}]$. For FM-MA that uses input-dependent joint-specific

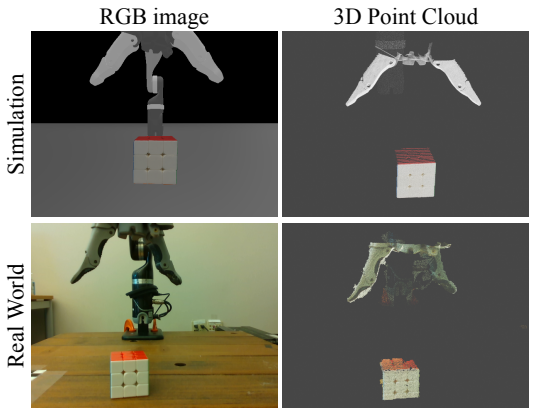



Figure 16: RGB images and 3D point clouds captured in both simulation and the real world. Colored point clouds for better illustration.

| Hyperparameters | Value |
|---|---|
| Optimizer | Adam |
| Discount ($\gamma$) | 0.95 |
| $\lambda$ in GAE | 0.95 |
| PPO clip range | 0.2 |
| Coefficient of the entropy loss term of PPO $c_{ent}$ | 0.0 |
| Advantage normalization | True |
| Reward normalization | True |
| Number of threads for collecting samples | 5 |
| Number of samples per PPO update | 40000 |
| Number of epochs per PPO update | 2 |
| Number of samples per minibatch | 330 |
| Gradient norm clipping | 0.5 |
| Max KL | 0.2 |
| Policy learning rate | 3e-4 (non FM-TG); 1e-4 (FM-TG) |
| Value learning rate | 3e-4 |
| Action MLP Last Layer Initialization | Zero-init |

Table 4: Hyperparameters for PPO.

weights to fuse action proposals from different frames, the MLP has dimension $[192, n \times m]$, where $n$ is the number of frames and $m$ is the dimension of action space. For FM-TG that uses Transformer to fuse features from different frames, the Transformer has 3 layers with hidden dimension 300 and feed-forward dimension 1024. For all network variants, the value head takes the concatenation of all visual features from all frames as input and passes through an MLP with dimensions $[192, 128, 1]$ to output value prediction.

In addition, we found that zero-initializing the last layer of MLP before action output along with the joint-specific weights in FM-MA to be very helpful for stabilizing agent training.

For each task, we train an agent for a fixed number of environment steps. Specifically, for OpenCabinetDoor, OpenCabinetDrawer, and MoveBucket, we train for 15 million steps. For PushChair, we train for 20 million steps. For PickObject, we train for 4 million steps. Success rates are calculated among 300 evaluation trajectories.

### S.5 More Details of Real-World Experiments

Fig. 16 shows the captured RGB images and point clouds in both simulation and the real world (by RealSense camera). For both simulation and the real-world environment, the ground points are removed using z-coordinate threshold or RANSAC, and the distant points are clipped. To reduce the sim-to-real gap, we only use xyz coordinates as our input point cloud feature, and we discard RGB colors.