FetchBot: Learning Generalizable Object Fetching in Cluttered Scenes via Zero-Shot Sim2Real

Weiheng Liu[1,3,4,*] **Yuxuan Wan**[2,3,*] **Jilong Wang**[2,3] **Yuxuan Kuang**[2] **Wenbo Cui**[1,3,4]
Xuesong Shi[3] **Haoran Li**[1] **Dongbin Zhao**[1] **Zhizheng Zhang**[3,4,†] **He Wang**[2,3,4,†]
[1]Institute of Automation, Chinese Academy of Sciences
[2]CFCS, School of Computer Science, Peking University
[3]Galbot [4]Beijing Academy of Artificial Intelligence

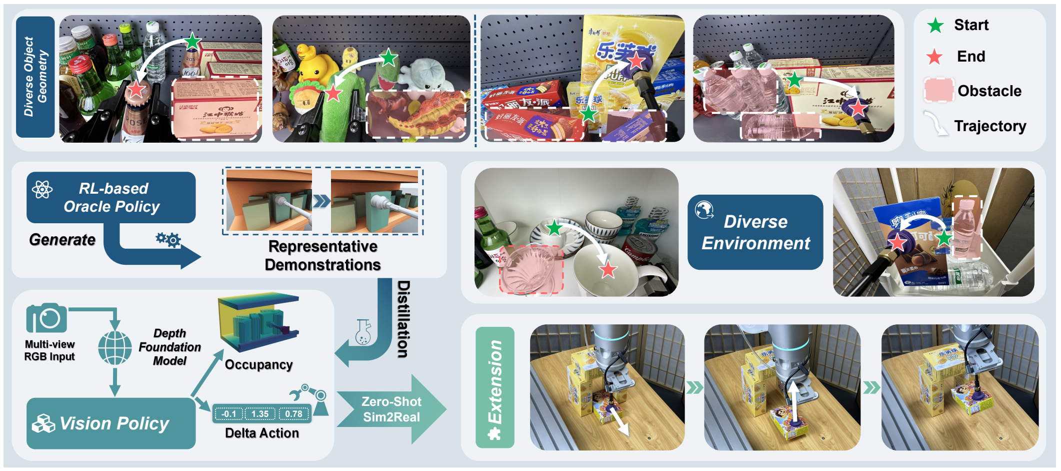


Figure 1: FetchBot is a sim-to-real framework achieving generalizable object fetching in cluttered scenes. Its systematic design enables policy generalization and sim-to-real transferability across diverse objects, varying layouts, and multiple end-effectors. Videos are on the ***project website***.

Abstract: Generalizable object fetching in cluttered scenes remains a fundamental and application-critical challenge in embodied AI. Closely packed objects cause inevitable occlusions, making safe action generation particularly difficult. Under such partial observability, effective policies must not only generalize across diverse objects and layouts but also reason about occlusion to avoid collisions. However, collecting large-scale real-world data for this task remains prohibitively expensive, leaving this problem largely unsolved. In this paper, we introduce FetchBot, a sim-to-real framework for this challenge. We first curate a large-scale synthetic dataset featuring 1M diverse scenes and 500k representative demonstrations. Based on this dataset, FetchBot employs a depth-conditioned method for action generation, which leverages structural cues to enable robust obstacle-aware action planning. However, depth is perfect in simulation but noisy in real-world environments. To address this sim-to-real gap, FetchBot predicts depth from RGB inputs using a foundation model and integrates local occupancy prediction as a pre-training task, providing a generalizable latent representation for sim-to-real transfer. Extensive experiments in simulation and real-world environments demonstrate FetchBot's strong zero-shot sim-to-real transfer, effective clutter handling, and adaptability to novel scenarios. In cluttered environments, it achieves an average real-world success rate of 89.95%, significantly outperforming prior methods. Moreover, FetchBot demonstrates excellent robustness in challenging

* Equal contributions; author order is arbitrary and does not reflect the level of contribution. Email: weihliu2002@gmail.com, yaser_wyx@163.com.
† Corresponding authors. Email: zhangzz@galbot.com, hewang@pku.edu.cn.

cases, such as fetching transparent, reflective, and irregular objects, highlighting its practical value. Project website: ***https://pku-epic.github.io/FetchBot/***.

Keywords: Generalizable Fetching, Sim2Real, Occlusion Handling

1 Introduction

Cluttered scenes are ubiquitous—from densely packed retail displays and disorganized warehouse racks to crowded kitchen cabinets—making reliable object fetching in such environments an essential capability for embodied AI applications. Densely arranged objects introduce severe occlusions and rich obstacles, requiring an effective fetching policy to reason about object geometry and obstacles to minimize collisions under partial observation while generalizing across diverse categories, layouts, and materials. These demands make generalizable fetching in cluttered scenes a largely unsolved problem.

For this task, some prior works [1, 48, 34, 2, 26] that employ heuristics or motion planning based on partial environment observation often fall short in highly cluttered scenes, especially when no collision-free path exists. Other works [38, 11] investigate learning-based methods, however they struggle to handle the generalization across diverse object arrangements, categories, geometries, and materials commonly encountered in real-world scenarios due to limited amount of real data [15, 12, 4]. In contrast, synthetic data offers a more efficient and scalable alternative. Some recent works [11, 38] propose to augment existing fetching datasets in simulation environments. However, the generated scenes are overly sparse, featuring widely spaced objects with minimal occlusion. Therefore, the policies fail to generalize to the densely cluttered environments encountered in the real world. Furthermore, sim-to-real gap persists. Several methods [22, 46, 31, 43] leverage depth and point-cloud inputs to bridge this gap, given its focus on geometry instead of texture [54]. Nevertheless, practical limitations still remain: real depth sensors often produce flying pixels around object boundaries [28], unreliable measurements on reflective and transparent surfaces, and substantial noise. These factors collectively undermine robust performance in complex real-world scenarios.

Along with generalization, fetching safety, which requires environmental impact minimization, is crucial since slight collisions may lead to hazards (illustrated in Fig. 1). Works [33, 10, 16, 21] model environments using scene voxel maps, typically constructed by projecting RGB views via sensed depth and camera extrinsics. This technique directly inherits the limitations of depth sensors, which are often unreliable in practice. Critically, it also produces incomplete voxel representations in heavily occluded regions, thereby losing essential geometric details.

To address the above challenges, we introduce FetchBot, a framework designed for generalizable and safe robotic fetching in occlusion-rich environments. To mitigate data scarcity, we develop a Unified Voxel-based Scene Generator (UniVoxGen) to synthesize millions of cluttered scenes efficiently, where objects are tightly packed and frequently occlude one another. Given that motion planning often fails to find collision-free trajectories in such scenes, we train a dynamics-aware oracle policy using reinforcement learning to generate representative demonstrations. We then leverage depth predicted by a foundation model [49] as an intermediate representation to bridge the sim-to-real gap. To tackle the incomplete scene understanding caused by heavy occlusion, we integrate occupancy prediction [42, 44, 3] as an auxiliary task to overcome perception limitations arising from heavy occlusion. This task encourages the model to infer occluded regions through spatial reasoning, leading to more complete scene understanding. Through extensive simulation and real-world experiments, our method outperforms existing approaches by a significant margin, achieving an average 89.95% success rate in real-world scenarios. This robustness also extends to challenging objects, including translucent, reflective, and irregular items.

In summary, we make the following contributions: 1) Generate a large-scale dataset comprising 1M diverse cluttered scenes using UniVoxGen and 500k demonstrations using a dynamics-aware oracle policy. This dataset serves as a foundation for developing generalizable fetching skills. 2) Introduce a depth-conditioned action generation policy. This policy leverages depth predictions from a foundation model to bridge the sim-to-real gap. Additionally, it integrates occupancy prediction as a pre-training task, enabling the model to infer information about occlusions and achieve comprehensive scene understanding. 3) Validate our method through extensive simulation and real-world experiments, demonstrating significant improvements in handling cluttered scenes, achieving zero-shot sim-to-real transfer, and generalizing across diverse scenarios.

2 Related Works

Robotic Fetching from Cluttered Scenes. Robotic fetching is a long-standing and fundamental challenge within robotic manipulation, attracting extensive research interest over the years [27]. A particularly demanding aspect of this challenge involves fetching objects from cluttered environments [6, 7, 23, 24, 51, 18, 26, 48, 1, 11, 53, 40]. While numerous studies [26, 48, 1] have focused on identifying initial picking or grasping points, the subsequent, critical retrieval stage is often under-emphasized. This phase necessitates careful maneuvering to minimize disturbance to surrounding objects, as even slight collisions can destabilize the scene. Recently, benchmarks like FetchBench [11] have begun to specifically target the complexities of object fetching. However, generating simulated environments that fully capture the high density and unpredictable nature of real-world clutter remains difficult, potentially limiting the direct applicability of models trained in such settings. Addressing this gap, our work employs a voxel-based method to generate simulated cluttered scenes that more realistically capture the complexity of real-world environments.

Sim2Real Transfer for 3D Visuomotor Policies. Currently, there are many 3D-based imitation learning policies [33, 10, 8, 52, 16, 45, 21] that utilize 3D observation data to mimic expert actions from demonstrations. However, these methods predominantly rely on real-world data to perform real-robot tasks, failing to fully leverage the potential of simulators. As a result, sim-to-real transfer for 3D visuomotor policies remains an under-explored topic. Most previous works [22, 46, 31, 43, 20] have employed point clouds as representations to achieve sim-to-real, but they still struggle to bridge the sim-to-real gap due to noise and inaccuracies, particularly at object edges and reflective surfaces in real-world point clouds captured by depth sensors. To further advance the field of sim-to-real research, inspired by approaches in autonomous driving [42, 3, 44] and computer vision foundation model [49, 9, 13, 32, 14], we propose utilizing predicted depth as an intermediate representation, leveraging its strong generalization ability to mitigate the sim-to-real gap. Furthermore, we employ a unified 3D representation to ensure consistent multi-view image fusion, thereby achieving complete perception and further minimizing the sim-to-real gap.

3 FetchBot

3.1 Overview

To enable generalizable and safe fetching skills, we introduce FetchBot, a framework that learns a generalizable fetching policy through zero-shot sim-to-real. We begin by efficiently generating realistic, densely cluttered scenes using a voxel-based method (§ 3.2), and collecting demonstrations from a dynamics-aware oracle policy (§ 3.3). Next, we leverage the depth predicted by a foundation model as the intermediate representation to bridge the sim-to-real gap (§ 3.4). Finally, we introduce an occupancy-based voxel representation to address the challenge of incomplete perception caused by heavy occlusion (§3.4).

3.2 Voxel-based Cluttered Scene Generator

Accurate scene generation mandates collision-free object placement. However, existing methods [38, 19, 47, 30, 50, 41] often rely on computationally intensive simulator-based collision checks or simulated object dropping. These techniques are prone to generating unrealistic configurations (see Appendix A for details) and significantly hinder both generation efficiency and scene validity, particularly in densely cluttered and occluded environments.

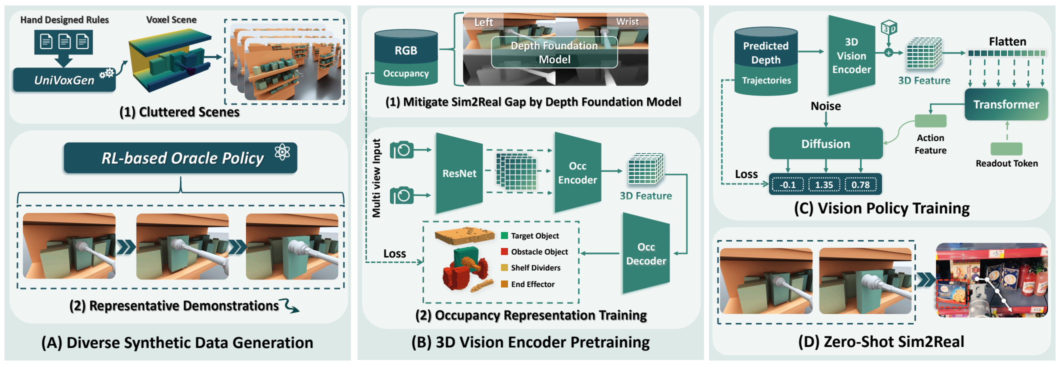


Figure 2: **FetchBot Pipeline.** In the (A) data generation stage, we use UniVoxGen to generate a diverse set of cluttered scenes and employ an RL-based Oracle policy to collect representative demonstrations. In the (B) 3D vision encoder pretraining stage, we first use the foundation model's predicted depth as an intermediate representation to mitigate the sim-to-real gap, then introduce an occupancy prediction task to learn a complete scene representation that can infer occluded regions. In the (C) vision policy training stage, we distill these expert demonstrations into a vision-based policy through imitation learning, which can achieve (D) zero-shot sim-to-real.

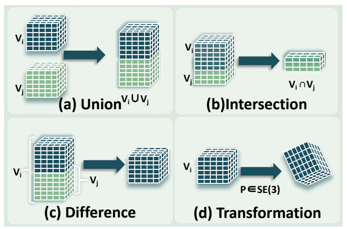


Figure 3: A set of fundamental voxel operations in UniVoxGen.

In contrast, our approach, UniVoxGen, operates within a unified voxel space established by initially voxelizing each object. As depicted in Fig. 3, we leverage fundamental voxel operations: *Union* (adding objects), *Intersection* (detecting collisions), *Difference* (removing objects), and *Transformation* (adjusting object poses). These operations are computationally lightweight and execute rapidly, facilitating the efficient generation of realistic scenes. Leveraging this efficiency, we employed UniVoxGen to create a large-scale dataset of 1 million cluttered scenes, which serve as training data for our oracle policy. Furthermore, the generated voxel scenes provide dense ground truth for training the occupancy prediction task.

3.3 Dynamics-Aware Oracle Policy

To generate demonstrations with minimal disturbance, we train an oracle policy using reinforcement learning. Through extensive interaction, this policy learns implicit environment dynamics, enabling it to understand the impact of potential actions and select optimal, low-impact maneuvers. A core component of this policy is a hierarchical scene encoder designed to capture both local object details and global scene context, allowing for effective representation of complex clutter.

Observations Encoding. The observation $\boldsymbol{o}_t$ of oracle policy $\boldsymbol{\pi}$ combines proprioception $\boldsymbol{p}_t$, the previous action $\boldsymbol{a}_{t-1}$, and a scene representation $\boldsymbol{z}_t$. Inspired by PointNet++ [29], we adopt a hierarchical network $\boldsymbol{f}_{scene}$. As depicted in Fig. 4, consider a scenario $\boldsymbol{S}$ comprising a set of objects, and each object is represented by a set of keypoints $\boldsymbol{K}_i = \{\boldsymbol{k}_{i,1}, \boldsymbol{k}_{i,2}, \ldots \boldsymbol{k}_{i,n}\}$. Our hierarchical process first extracts local features $\boldsymbol{z}_{local}^{(i)}$ for each object independently using a network $\boldsymbol{f}_{local}$: $\boldsymbol{z}_{local}^{(i)} = \boldsymbol{f}_{local}(K_i)$. Subsequently, these local features are aggregated by a global network $\boldsymbol{f}_{global}$ to produce the final scene representation $\boldsymbol{z} = \boldsymbol{z}_{global}$: $\boldsymbol{z}_{global} = \boldsymbol{f}_{global}(\{\boldsymbol{z}_{local}^{(1)}, \ldots, \boldsymbol{z}_{local}^{(N)}\})$. This global feature $\boldsymbol{z}$ serves as our comprehensive scene representation. The networks $\boldsymbol{f}_{local}$ and $\boldsymbol{f}_{global}$ each comprise a two-layer MLP followed by max pooling. This design enables permutation invariance, there-

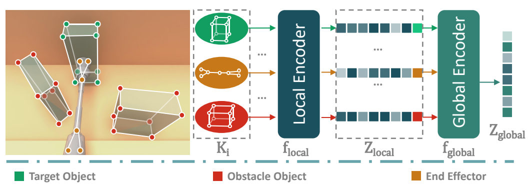


Figure 4: **Scene Encoder Network.** The network employs a hierarchical architecture, where a local network extracts object features and a global network captures scene-level structure and context.

fore enhancing robustness. This hierarchical architecture could preserve detailed object information alongside global scene understanding.

Reward. Our reward function combines task completion, behavioral constraints, and an environment change penalty based on per-step obstacle movement m_{step} (translation and rotation): $r = \lambda_{task}r_{task} + \lambda_{cons}r_{cons} + \lambda_{env}r_{env}$. Task reward is given for successful fetching, but only if the total obstacle change per episode $m_{episode}$ is less than σ_m, a success threshold. We employ a curriculum for σ_m, starting high to facilitate initial task completion and gradually decreasing it to enforce precision and minimal environmental impact later in training. Detailed reward design is provided in Appendix B.

3.4 Vision-based Imitation Learning

Intermediate Depth Representation for Sim-to-Real. Depth maps offer superior spatial information compared to RGB images, which is crucial for robotic fetching. Nonetheless, direct sim-to-real policy transfer with depth is difficult due to depth sensor limitations like edge inaccuracies, poor transparency handling, and substantial noise. To address this, we utilize a sim-to-real-friendly intermediate depth representation derived from a depth foundation model. We use DepthAnything [49] to map both simulation RGB images P_s and real-world RGB images P_r to a shared depth space D_f. This intermediate representation ensures the policy always receives input within the same input space D_f. Consequently, this design allows FetchBot to maintain consistent action predictions across domains and significantly improve its sim-to-real capability.

Completion of Occluded Regions. Planning safe trajectories in cluttered scenes is challenging due to heavy occlusion hindering complete perception. As depicted in Fig. 2 (B), we address this by introducing semantic occupancy prediction as an auxiliary task. After obtaining multi-view predicted depth maps from the DepthAnything, we extract feature maps $I = \{i_i\}_{i=1}^{N}$ using a ResNet-50 backbone. These features are subsequently aggregated into a 3D latent space using a multi-view feature fusion mechanism. Specifically, within the 3D vision encoder, we define a set of learnable local 3D-grid queries $Q \in \mathbb{R}^{C\times H\times W\times Z}$, where H, W, and Z denote the query grid dimensions. For each 3D query point p, we project it into the 2D feature maps $\mathcal{F}^{2D}$ according to known camera parameters, utilizing only information from the valid views V_{hit}. A deformable cross-attention (DCA) mechanism then aggregates local 2D features around the projected points:

$$\text{DCA}(q_p, \mathcal{F}^{2D}) = \frac{1}{|V_{hit}|}\sum_{t\in V_{hit}} \text{DA}(q_p, \mathcal{P}(p,t), \mathcal{F}_t^{2D}),$$

where q_p is the feature associated with 3D point $p = (x, y, z)$, $\mathcal{P}(p,t)$ is the projection function, $\mathcal{F}_t^{2D}$ is the 2D feature map from hitted view t, and DA denotes deformable attention [55]. The aggregated 3D features are further refined using 3D convolutions (see Appendix C for more details).

Within a local 3D volume around the end-effector, referred to as the region of interest (ROI), we predict semantic occupancy, distinguishing object classes (target, obstacle, robot) instead of simply identifying occupied or free spaces. This semantic representation is crucial for identifying the target object in cluttered scenes. Local prediction within ROI improves both computational efficiency and generalization by exposing the network to diverse partial observations (irregular shape due to cropping). Additionally, the network uses multi-view predicted depth maps as input to predict occupancy in unobserved regions, and the model is supervised with complete scene occupancy (generated by UniVoxGen). This auxiliary task enables the model to infer the full structure from limited cues, enhancing robustness to occlusions in cluttered scenes.

As illustrated in Fig. 2 (C), building upon the 3D semantic occupancy prediction, our policy translates this scene representation $\mathcal{F}^{3D}$ into executable actions through a transformer-based architecture [37, 17, 39]. Specifically, we augment the 3D features with learnable position embeddings $P^{3D} \in \mathbb{R}^{C\times H\times W\times Z}$ before flattening them into feature vectors $V^{3D} \in \mathbb{R}^{C}$. A learnable readout token queries and aggregates these features to produce action features $\mathcal{F}_A$. Finally, a lightweight diffusion head [5] denoises Gaussian noise a^K into the predicted action a^0, conditioned on $\mathcal{F}_A$.

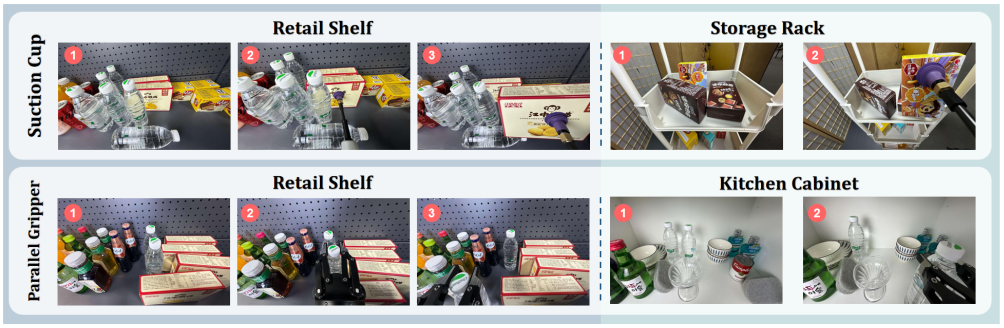


Figure 5: **Qualitative Results in the real world.** We evaluate our system in diverse and cluttered scenes containing a variety of objects and environments. The upper row shows a suction cup, and the bottom row shows the parallel gripper.

End-effector	**Suction Cup**			**Parallel Gripper**		
Method	**Success Rate (%) ↑**	**Translation (cm) ↓**	**Rotation (rad) ↓**	**Success Rate (%) ↑**	**Translation (cm) ↓**	**Rotation (rad) ↓**
Heuristc	54.98	6.71	0.91	63.43	0.41	0.71
CuRobo	68.26	3.96	0.63	73.05	0.24	0.49
AIT*	62.52	5.86	0.75	67.87	0.35	0.63
Oracle	**85.60**	**1.47**	**0.29**	**94.11**	**0.13**	**0.02**
RGB (DP)	70.35	4.97	0.55	75.85	0.24	0.13
Point Cloud (DP3)	72.37	4.49	0.76	80.45	0.21	0.16
Raw Depth	71.44	4.49	0.56	71.44	0.17	0.09
Predicted Depth	61.21	6.54	0.76	68.37	0.38	0.12
RGB-D Voxel	71.38	4.65	0.62	79.73	0.22	0.10
Occupancy (Ours)	**81.46**	**2.78**	**0.36**	**91.02**	**0.13**	**0.06**

Table 1: **Simulation Results.** Compares the performance of different methods across two end effectors (suction cup and parallel gripper) in terms of success rate, translation disturbance, and rotation disturbance. Our method outperforms all baselines (except for the oracle), achieving the highest success rate and the lowest disturbance in translation and rotation.

Two-Stage Policy Learning Framework. We begin by pretraining the 3D vision encoder on UniVoxGen's complete occupancy data, optimizing cross-entropy and scene-class affinity losses [3] to establish robust scene understanding. In the second stage, we freeze the pre-trained vision encoder. Concurrently, the policy head (comprising transformer and diffusion components) is trained via imitation learning. This learning process minimizes the mean squared error (MSE) between the predicted noise and the actual noise applied to the oracle actions.

4 Experiments

In this section, we conduct extensive experiments to validate the effectiveness of our method. Our primary evaluations focus on shelf environments, which present the most significant challenges for fetching tasks. First, we compare our method with several baselines in the Isaac Gym [25] simulator. Next, we deploy the system in a replicated retail store to confirm the sim-to-real performance. Finally, we conduct ablation studies to demonstrate the contribution of each key component to the sim2real transfer. We use two end effectors, a parallel gripper for bottle-shaped items and a suction cup for box-shaped items, to demonstrate our method's flexibility.

4.1 Simulation Experiment

In simulation experiments, we utilize 3000 densely constructed scenes to assess safe fetching performance under occlusion. We further investigate the efficacy of different representation methods for fetching in these cluttered environments by modifying the input modality module.

Metrics. Our simulation evaluations assess two key metrics: (1) environmental impact, measured by total obstacle displacement (both translational and rotational), evaluates fetching-induced disturbances, and (2) success rate, requiring target fetching with obstacle displacement kept under a 3 cm threshold, which we determined to be an acceptable tolerance during fetching.

Baselines. We compare FetchBot with three series of methods: (1) heuristic-based methods, utilizing rules to generate trajectories composed of straight-line paths. (e.g., lift-then-extract), (2) motion planning-based methods (CuRobo [36] and AIT* [35]), which find collision-free paths for fetching via motion planners, and (3) learning-based methods, including the original RGB-based diffusion policy (DP [5]), 3D point cloud-based DP3 [52], RGB-D voxel-based diffusion policy, raw depth image-based diffusion policy, and DepthAnything-based diffusion policy, all these baselines employ the same diffusion denoising process for action generation (see Appendix D for more details).

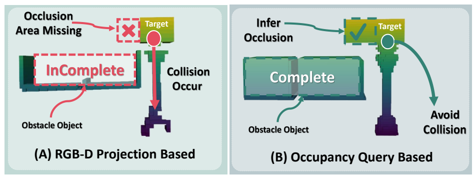


Figure 6: **Comparison in simulation. (A)** RGB-D voxel method misses crucial geometric details due to occlusion, leading to collision during fetching. **(B)** Our method can infer the occluded region, enabling successful collision avoidance.

Results and Analysis. As shown in Table 1, heuristic-based methods exhibit the poorest performance due to their complete disregard for environmental information and reliance on predefined trajectories, resulting in frequent collisions. Motion planning-based approaches also demonstrate limited effectiveness, failing to reliably generate collision-free paths in cluttered scenes as their planners account only for static geometric configurations while neglecting the dynamic consequences of fetching operations. Conversely, our oracle policy learns underlying environmental dynamics through extensive interactions, enabling it to select actions that minimize environmental change. Learning-based methods achieve substantially superior performance, though with notable constraints: RGB (DP) policy encounters obstacle avoidance difficulties stemming from absent spatial geometric information, while point cloud, depth-based, and RGB-D voxel methods, despite their enhanced scene representations, are hindered by partial observability induced by severe occlusions, ultimately diminishing success rates and environmental impact metrics. As shown in Fig. 6 (A), due to occlusion, the RGB-D voxel method fails to capture the hidden part of the target, resulting in a naive extraction attempt and a subsequent collision with obstacles. Besides struggling with heavy occlusion, the DepthAnything-based method's performance is further hindered by its scale-ambiguous depth predictions [49], yielding only relative depth, inadequate for true geometric representation. By contrast, FetchBot's occupancy representation effectively integrates multi-view spatial information and uses contextual reasoning to infer geometric details of occluded areas, as depicted in Fig. 6 (B). This approach leads to a more complete and structured 3D scene understanding, which facilitates object fetching in complex environments, ultimately achieving a higher success rate of 81.46% for suction cup and 91.02% for parallel gripper. Furthermore, FetchBot excels in the environmental impact metric, meaning it effectively considers how each action affects nearby objects and chooses actions that minimize disturbance.

4.2 Real World Experiment

Extensive real-world experiments validated each method's generalization and zero-shot sim-to-real transfer across diverse scenarios (varied objects, challenging materials, environments) using 15 trials with both suction and gripper end-effectors (All scenes are shown in Appendix E).

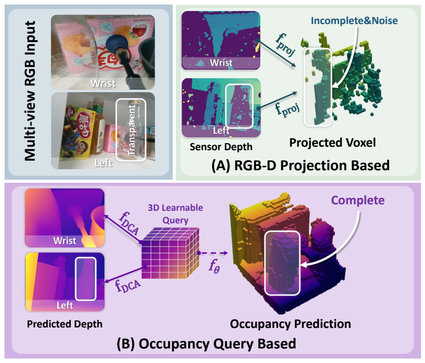


Figure 7: Comparison in the real world. Direct RGB-D voxelization often yields incomplete scenes in real-world settings, while our occupancy method (Occ) generates more complete representations.

Metrics. Due to the unavailability of precise environmental measurements in real-world settings, the success rate serves as the only evaluation metric. A real-world trial is counted as successful only if the target is acquired and no surrounding objects exhibit visible displacement during the fetch.

Baselines. Our real-world experimental evaluation focuses exclusively on learning-based approaches, employing identical representation modalities to those used in simulation (RGB, point clouds, raw depth, DepthAnything, and RGB-D voxel).

Results and Analysis. Fig. 8 presents the zero-shot sim-to-real performance of each method in the real world. Due to differences in object textures, lighting, and other factors between simulation and reality, the RGB (DP) still underperforms despite extensive domain randomization. Additionally, the impact of depth noise and the inherent limitations of depth sensors (e.g., poor performance on transparent and specular materials) limit the effectiveness of point cloud and raw depth-based methods in zero-shot sim-to-real transfer, as illustrated in Fig. 7 (A). While the DepthAnything-based method uses an intermediate representation to predict depth from both sim and real-world images, partially addressing the sim-to-real gap, its predicted depth is scale-ambiguous [49], which ultimately limits the method's overall performance. Moreover, limited observations due to cluttered scenes further restrict its deployment. In contrast, FetchBot leverages predicted depth from a foundation model to bridge the sim-to-real gap and handle challenging objects like transparent or reflective ones. It also uses an occupancy-based representation to infer occluded regions, providing complete and structured geometry for better spatial understanding, as shown in Fig. 7 (B). By focusing on local occupancy and semantics within ROI and extensively randomizing object placements there during training, the system generalizes effectively by exposing the policy to a wide variety of local geometries. Its success in varied tasks, including tasks like tabletop fetching, as well as drawer extraction without collision, demonstrates broad applicability beyond shelf setting (shown in Fig. 9).

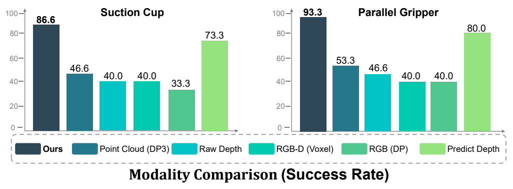


Figure 8: Modalities comparison in the real world.

4.3 Ablation Study for Sim2Real

We perform real-world ablation studies to systematically evaluate how individual modules in FetchBot contribute to successful sim-to-real transfer and overall fetching performance. As shown in Table 2, FetchBot w/o DepthAnything significantly reduces the real-world success rate (from 86.60% to 60.00%), confirming that predicted depth is crucial for mitigating the visual sim-to-real gap. Removing the occupancy representation also degrades performance (to 73.33%), demonstrating its necessity for inferring the occluded regions and alleviating the associated perception limitations. Using only RGB inputs (removing both) results in the lowest success rate (33.33%), highlighting the synergistic benefit of the two modules for reliable fetching in challenging real-world conditions.

DepthAnything	Occupancy	Success Rate (%) ↑
✓	✓	**86.60**
✓	×	73.33
×	✓	60.00
×	×	33.33

Table 2: Ablation study for sim2real.

5 Conclusion and Limitation

Conclusion. This paper introduces FetchBot, a novel sim-to-real framework tackling the challenge of object fetching in cluttered scenes. By leveraging a large-scale synthetic dataset, depth-conditioned action generation, and techniques to bridge the sim-to-real gap (including RGB-to-depth prediction and occupancy pretraining), FetchBot demonstrates strong zero-shot transfer capabilities. Extensive experiments validate its superior performance in effectively managing clutter and handling challenging objects (e.g., transparent, reflective and irregular), significantly outperforming previous methods and showcasing its practical value for complex, real-world fetching tasks.

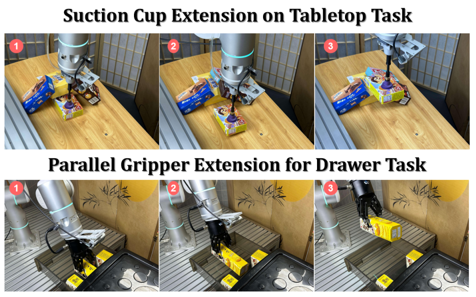

Suction Cup Extension on Tabletop Task
1
2
3
Parallel Gripper Extension for Drawer Task
1
2
3

Figure 9: Real-robot extension experiments showing successful fetching from cluttered tabletop (suction) and drawer (parallel gripper) settings.

Limitation. FetchBot exhibits limitations despite its performance. Fetchbot exhibits certain limitations when encountering tricky scenarios. First, in scenarios with significant occlusion, the policy may output complex actions to avoid collisions. However, such maneuvers can cause the robot arm to exceed its joint limits. Second, single-arm grasping of large-volume objects is inherently difficult, suggesting a potential need for dual-arm collaboration in future work. Finally, the challenge with fully occluded targets lies in their initial unreachability. Accessing them requires advanced reasoning to clear obstructions. Ideally, this reasoning would also encompass restoring the scene after the task (e.g., returning moved objects). Developing such comprehensive reasoning capabilities may be a key objective for future work. (Please refer to Appendix F for more details)

Acknowledgments

We would like to express our sincere gratitude to Feng Zhu for his valuable contributions to the real-world experiments, particularly for designing the prototype of our suction cup. We also thank Zhennan Jiang, Yupeng Zheng, and Jiangran Lv for their helpful discussions and assistance with proofreading.

References

[1] Soofiyan Atar, Yi Li, Markus Grotz, Michael Wolf, Dieter Fox, and Joshua Smith. Optigrasp: Optimized grasp pose detection using rgb images for warehouse picking robots. *arXiv preprint arXiv:2409.19494*, 2024.

[2] Max Bajracharya, James Borders, Richard Cheng, Dan Helmick, Lukas Kaul, Dan Kruse, John Leichty, Jeremy Ma, Carolyn Matl, Frank Michel, et al. Demonstrating mobile manipulation in the wild: A metrics-driven approach. *arXiv preprint arXiv:2401.01474*, 2024.

[3] Anh-Quan Cao and Raoul De Charette. Monoscene: Monocular 3d semantic scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3991–4001, 2022.

[4] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24185–24198, 2024.

[5] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[6] Nikolaus Correll, Kostas E Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M Romano, and Peter R Wurman. Analysis and observations from the first amazon picking challenge. *IEEE Transactions on Automation Science and Engineering*, 15(1):172–188, 2016.

[7] Clemens Eppner, Sebastian Höfer, Rico Jonschkowski, Roberto Martín-Martín, Arne Sieverling, Vincent Wall, and Oliver Brock. Lessons from the amazon picking challenge: Four aspects of building robotic systems. In *Robotics: science and systems*, volume 12, 2016.

[8] Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3d: Infinite resolution action detection transformer for robotic manipulation. *arXiv preprint arXiv:2306.17817*, 2023.

[9] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3828–3838, 2019.

[10] Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pages 694–710. PMLR, 2023.

[11] Beining Han, Meenal Parakh, Derek Geng, Jack A Defay, Gan Luyang, and Jia Deng. Fetchbench: A simulation benchmark for robot fetching. *arXiv preprint arXiv:2406.11793*, 2024.

[12] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[13] Wenbo Hu, Xiangjun Gao, Xiaoyu Li, Sijie Zhao, Xiaodong Cun, Yong Zhang, Long Quan, and Ying Shan. Depthcrafter: Generating consistent long depth sequences for open-world videos. *arXiv preprint arXiv:2409.02095*, 2024.

[14] Muhammad Zubair Irshad, Sergey Zakharov, Vitor Guizilini, Adrien Gaidon, Zsolt Kira, and Rares Ambrus. Nerf-mae: Masked autoencoders for self-supervised 3d representation learning for neural radiance fields. In *European Conference on Computer Vision*, pages 434–453. Springer, 2024.

[15] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[16] Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

[17] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[18] Juncheng Li and David J Cappelleri. Sim-suction: Learning a suction grasp policy for cluttered environments using a synthetic benchmark. *IEEE Transactions on Robotics*, 2023.

[19] Yitong Li, Ruihai Wu, Haoran Lu, Chuanruo Ning, Yan Shen, Guanqi Zhan, and Hao Dong. Broadcasting support relations recursively from local dynamics for object retrieval in clutters. In *Robotics: Science and Systems*, 2024.

[20] Haotong Lin, Sida Peng, Jingxiao Chen, Songyou Peng, Jiaming Sun, Minghuan Liu, Hujun Bao, Jiashi Feng, Xiaowei Zhou, and Bingyi Kang. Prompting depth anything for 4k resolution accurate metric depth estimation. *arXiv preprint arXiv:2412.14015*, 2024.

[21] I Liu, Chun Arthur, Sicheng He, Daniel Seita, and Gaurav Sukhatme. Voxact-b: Voxel-based acting and stabilizing policy for bimanual manipulation. *arXiv preprint arXiv:2407.04152*, 2024.

[22] Jiangran Lyu, Yuxing Chen, Tao Du, Feng Zhu, Huiquan Liu, Yizhou Wang, and He Wang. Scissorbot: Learning generalizable scissor skill for paper cutting via simulation, imitation, and sim2real. *arXiv preprint arXiv:2409.13966*, 2024.

[23] Jeffrey Mahler and Ken Goldberg. Learning deep policies for robot bin picking by simulating robust grasping sequences. In *Conference on robot learning*, pages 515–524. PMLR, 2017.

[24] Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26):eaau4984, 2019.

[25] Viktor et al. Makoviychuk. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv:2108.10470*, 2021.

[26] Michael Murray, Abhishek Gupta, and Maya Cakmak. Learning to grasp in clutter with interactive visual failure prediction. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 18172–18178. IEEE, 2024.

[27] Rhys Newbury, Morris Gu, Lachlan Chumbley, Arsalan Mousavian, Clemens Eppner, Jürgen Leitner, Jeannette Bohg, Antonio Morales, Tamim Asfour, Danica Kragic, et al. Deep learning approaches to grasp synthesis: A review. *IEEE Transactions on Robotics*, 39(5):3994–4015, 2023.

[28] Sang-Ho Park, Ga-Rin Park, and Kwang-Ryul Baek. Edge bleeding artifact reduction for shape from focus in microscopic 3d sensing. *Sensors*, 23(20):8602, 2023.

[29] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[30] Yaoyao Qian, Xupeng Zhu, Ondrej Biza, Shuo Jiang, Linfeng Zhao, Haojie Huang, Yu Qi, and Robert Platt. Thinkgrasp: A vision-language system for strategic part grasping in clutter. *arXiv preprint arXiv:2407.11298*, 2024.

[31] Yuzhe Qin, Binghao Huang, Zhao-Heng Yin, Hao Su, and Xiaolong Wang. Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation. In *Conference on Robot Learning*, pages 594–605. PMLR, 2023.

[32] Jiahao Shao, Yuanbo Yang, Hongyu Zhou, Youmin Zhang, Yujun Shen, Matteo Poggi, and Yiyi Liao. Learning temporally consistent video depth from video diffusion priors. *arXiv preprint arXiv:2406.01493*, 2024.

[33] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[34] Max Spahn, Corrado Pezzato, Chadi Salmi, Rick Dekker, Cong Wang, Christian Pek, Jens Kober, Javier Alonso-Mora, C Hernandez Corbato, and Martijn Wisse. Demonstrating adaptive mobile manipulation in retail environments. *Proceedings of the Robotics: Science and System XX*, 2024.

[35] Marlin P Strub and Jonathan D Gammell. Adaptively informed trees (ait*): Fast asymptotically optimal path planning through adaptive heuristics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3191–3198. IEEE, 2020.

[36] Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, et al. Curobo: Parallelized collision-free robot motion generation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8112–8119. IEEE, 2023.

[37] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[38] Kentaro Wada, Stephen James, and Andrew J Davison. Safepicking: Learning safe object extraction via object-level mapping. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10202–10208. IEEE, 2022.

[39] Chenxi Wang, Hongjie Fang, Hao-Shu Fang, and Cewu Lu. Rise: 3d perception makes real-world robot imitation simple and effective. *arXiv preprint arXiv:2404.12281*, 2024.

[40] Jilong Wang, Javokhirbek Rajabov, Chaoyi Xu, Yiming Zheng, and He Wang. Quadwbg: Generalizable quadrupedal whole-body grasping. *arXiv preprint arXiv:2411.06782*, 2024.

[41] Yongliang Wang and Hamidreza Kasaei. Learning dual-arm push and grasp synergy in dense clutter. *arXiv preprint arXiv:2412.04052*, 2024.

[42] Yuqi Wang, Yuntao Chen, Xingyu Liao, Lue Fan, and Zhaoxiang Zhang. Panoocc: Unified occupancy representation for camera-based 3d panoptic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17158–17168, 2024.

[43] Zifan Wang, Ziqing Chen, Junyu Chen, Jilong Wang, Yuxin Yang, Yunze Liu, Xueyi Liu, He Wang, and Li Yi. Mobileh2r: Learning generalizable human to mobile robot handover exclusively from scalable and diverse synthetic data. *arXiv preprint arXiv:2501.04595*, 2025.

[44] Yi Wei, Linqing Zhao, Wenzhao Zheng, Zheng Zhu, Jie Zhou, and Jiwen Lu. Surroundocc: Multi-camera 3d occupancy prediction for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21729–21740, 2023.

[45] Zhou Xian, Nikolaos Gkanatsios, Theophile Gervet, Tsung-Wei Ke, and Katerina Fragkiadaki. Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In *7th Annual Conference on Robot Learning*, 2023.

[46] Pengwei Xie, Rui Chen, Siang Chen, Yuzhe Qin, Fanbo Xiang, Tianyu Sun, Jing Xu, Guijin Wang, and Hao Su. Part-guided 3d rl for sim2real articulated object manipulation. *IEEE Robotics and Automation Letters*, 2023.

[47] Kechun Xu, Shuqi Zhao, Zhongxiang Zhou, Zizhang Li, Huaijin Pi, Yifeng Zhu, Yue Wang, and Rong Xiong. A joint modeling of vision-language-action for target-oriented grasping in clutter. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11597–11604. IEEE, 2023.

[48] Boling Yang, Soofiyan Atar, Markus Grotz, Byron Boots, and Joshua Smith. Dynamo-grasp: Dynamics-aware optimization for grasp point detection in suction grippers. In *Conference on Robot Learning*, pages 2096–2112. PMLR, 2023.

[49] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv preprint arXiv:2406.09414*, 2024.

[50] Yuxiang Yang, Jiangtao Guo, Zilong Li, Zhiwei He, and Jing Zhang. Ground4act: Leveraging visual-language model for collaborative pushing and grasping in clutter. *Image and Vision Computing*, 151:105280, 2024.

[51] Kuan-Ting Yu, Nima Fazeli, Nikhil Chavan-Dafle, Orion Taylor, Elliott Donlon, Guillermo Diaz Lankenau, and Alberto Rodriguez. A summary of team mit's approach to the amazon picking challenge 2015. *arXiv preprint arXiv:1604.03639*, 2016.

[52] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *ICRA 2024 Workshop on 3D Visual Representations for Robot Manipulation*, 2024.

[53] Jiazhao Zhang, Nandiraju Gireesh, Jilong Wang, Xiaomeng Fang, Chaoyi Xu, Weiguang Chen, Liu Dai, and He Wang. Gamma: Graspability-aware mobile manipulation policy learning based on online grasping pose fusion. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1399–1405. IEEE, 2024.

[54] Xiaoshuai Zhang, Rui Chen, Ang Li, Fanbo Xiang, Yuzhe Qin, Jiayuan Gu, Zhan Ling, Minghua Liu, Peiyu Zeng, Songfang Han, et al. Close the optical sensing domain gap by physics-grounded active stereo sensor simulation. *IEEE transactions on robotics*, 39(3):2429–2447, 2023.

[55] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

Appendix

A UniVoxGen

A.1 Scene Generation Method

A diverse and realistic set of scenes is crucial for sim-to-real transfer, requiring an effective scene generation method. Creating a valid scene involves determining the pose for each object and ensuring there is no penetration among any of them. Therefore, during scene generation, we need to check for collisions between different objects. Previous approaches [6, 3, 8, 4, 10, 7] for generating cluttered scenes typically depend on computationally intensive collision detection mechanisms. These methods require importing objects into a simulator and executing a simulation step to detect potential collisions, a process known to be computationally expensive. This inefficiency is particularly pronounced in cluttered scenes, where the dense packing of objects inherently leads to frequent collisions requiring verification. Additionally, some other methods create cluttered layouts by simulating objects dropped from the air, which may result in unstable and unrealistic scene configurations. Our method, UniVoxGen, is specifically designed for fast and realistic scene generation in voxel space. It accelerates the generation process by performing efficient collision checks in voxel space and produces realistic scene layouts using a set of carefully crafted hand-designed rules.

We begin by providing formal definitions for key elements in the voxel space. Let $V^o = \{V_1^o, V_2^o, \dots, V_N^o\}$ represent the voxel representation of a set of objects, and $V^s = \{V_1^s, V_2^s, \dots, V_N^s\}$ represent the voxel representation of the scene. We define a set of operational primitives in voxel space for manipulating voxels. Specifically, $V_i \cup V_j$ denotes the union operation, which combines two voxel sets and is commonly used to add an object's voxels into the scene. $V_i \cap V_j$ denotes the intersection operation, which retrieves the intersection of two voxel sets and is used to detect potential collisions when adding a new object. $V_i - V_j$ denotes the difference operation, which removes the overlapping portion of V_i with V_j, typically used to remove an object from the scene. Finally, $T(V_i, P), P \in SE(3)$ represents a transformation of a voxel V_i in $SE(3)$ space, commonly used to change the pose of the object. Here, P is a transformation matrix in $SE(3)$ that combines rotation and translation.

Algorithm 1 Scene Generation Algorithm

```
Require: Number of scenes $N$
Require: Max objects per scene $K$
Require: A set of objects $\mathcal{O}$
 1: for scene $1:N$ do
 2:     Initialize scene voxel $V^s = \{\}$
 3:     Sample a target object $O^{tar}$
 4:     Sample a pose $P$ in SE(3) for $O^{tar}$
 5:     Apply $T(V_i, P)$ to transform the target object
 6:     Apply $V_{O^{tar}} \cup V^s$ to add the target object to the scene
 7:     Sample number of obstacle objects $k \sim [1, \dots, K]$
 8:     for obstacle $O_i^{obs}$ $1:k$ do
 9:         Sample a pose $P$ in SE(3) for $O_i^{obs}$
10:         Apply $T(V_i, P)$ to transform the obstacle object
11:         while $V_{O_i^{obs}} \cap V^s$ do
12:             Sample a new pose $P$ in SE(3) for $O_i^{obs}$
13:             Apply $T(V_i, P)$ to transform the obstacle object
14:         end while
15:         Apply $V_{O_i^{obs}} \cup V^s$ to add the obstacle object to the scene
16:     end for
17:     Save the pose $P$ of each object in the scene
18: end for
```

Based on the previously defined key elements and operation primitives, we further design a set of generation rules $R = \{R_1, R_2, \dots, R_N\}$. UniVoxGen uses these rules to generate diverse cluttered scenes. These scenes may include unsolvable cases, where it is impossible to retrieve the target

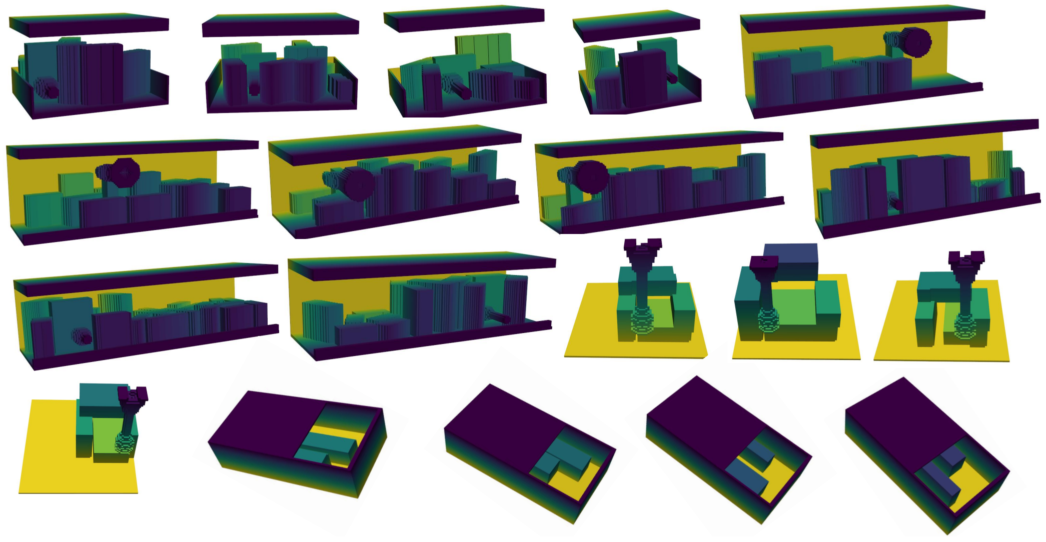


Figure 10: Generated cluttered scenes by UniVoxGen, including shelf, tabletop, drawer, and storage rack environments.

object without colliding with any obstacles. It is worth noting that the inclusion of unsolvable cases is intended to better simulate real-world scenarios, as such situations can occur in practice. The procedure for generating these cluttered scenes is outlined in Algorithm 1. It should be noted that, given the complexity of the various scene generation rules, the steps presented here represent a simplified version of our scene generation rules. The detailed generation rules will be made available in the released source code. Finally, we use UniVoxGen to generate ***1 million*** cluttered scenes, which are then utilized as training scene data for a oracle policy. It takes 12 hours on the workstation equipped with 8 RTX 4090s to generate these 1 million scenes.

A.2 Scene Generation Result

We utilize UniVoxGen to generate a large number of cluttered environments. Our initial focus is on cluttered shelf environments. Such environments require not only clutter but also a certain degree of structured arrangement to be realistic. On a shelf, objects cannot be arbitrarily piled together, but instead need to exhibit some regularity in their placement to simulate realistic shelf settings. Additionally, we generate cluttered environments for tabletop, drawer, and rack settings to support the needs of extension experiments, as shown in Fig. 10.

B Oracle Policy Training

B.1 Reward Function

Our reward function combines task completion r_{task}, behavioral constraints r_{cons}, and an environment change penalty r_{env}.

Task Reward. The task reward formulation imposes a dual requirement: the target object must be successfully fetched, and the resulting environmental impact, measured over the entire episode, must be limited. This impact is quantified by the total translation (m_{trans}) and total rotation (m_{rot}) accumulated across all obstacles during the episode. Both metrics must remain below their respective predefined thresholds: m_{trans} must be less than the translation threshold σ_{trans}, and m_{rot} must be less than the rotation threshold σ_{rot}. We use a curriculum learning method for the thresholds (σ_{trans} and σ_{rot}), gradually decreasing their values during training. Specifically, σ_{trans} is set according to the sequence [0.03, 0.015, 0.01, 0.005, 0.0] m, and σ_{rot} follows the sequence [0.4, 0.2, 0.16, 0.1, 0.0] radians. The transition to the next value in each sequence occurs once the policy's performance saturate at the current stage.

Behavioral Constraints Reward. We apply certain behavioral constraints to the flying end-effector to make its behavior more naturalistic.

We use r_{action_range} to constrain the policy's output from becoming too large. That is:

$$r_{action\_range} = \begin{cases} -\lambda_{action\_range} \exp\left((|a|-3)^2\right) & \text{if } |a| > 3 \\ 0 & \text{if } |a| \le 3 \end{cases}$$

To ensure smoother and more continuous actions, we introduce a term, r_{action_rate}, that penalizes large differences between consecutive pose outputs of the policy ($a_{last}, a_{current}$):

$$r_{action\_rate} = -\lambda_{action\_rate} \exp\left((a_{last} - a_{current})^2\right)$$

To prevent fetching the object in singular poses, which could violate robot arm joint limits, we incentivize maintaining the target object close to its initial pose (p_{init}). This objective is achieved using the r_{pose} penalty term, calculated as:

$$r_{pose} = -\lambda_{pose} \exp\left(p_{curr} - p_{init}\right)$$

To prevent collisions with shelf barriers, we introduce a penalty term, $r_{penetration}$. If a collision, i.e., penetration, occurs, the target object experiences a significant acceleration, denoted as a_{ccel}. This acceleration is then used to compute the penalty as follows:

$$r_{penetration} = -\lambda_{pen} \exp\left(a_{ccel}^2\right)$$

To encourage moving the target towards its designated goal position, we employ a reward component, r_{target_move}, to guide the policy's behavior. This reward incentivizes movement towards the goal and penalizes movement away from it. Let d_{last} denote the distance from the target to the goal position at the previous timestep, and d_{curr} be the current distance to the goal. The term $d_{last} - d_{curr}$ thus represents the progress made towards the goal in the current step, a positive value indicates movement closer to the goal. We use v represents target's velocity and dt is the duration of the timestep, the reward is defined as:

$$r_{target\_move} = \lambda_{target\_move} \cdot \frac{d_{last} - d_{curr}}{v \cdot dt}$$

Environment Change Reward. To incentivize the policy to minimize its impact on the environment, we introduce penalties based on the total displacement of all obstacles within each timestep. Specifically, we consider the total translational displacement (m_{trans_step}) and the total rotational displacement (m_{rot_step}) of all obstacles. These penalties are computed as follows:

$$r_{trans\_step} = -\lambda_{trans\_step} \exp\left(m_{trans\_step}^2\right)$$

$$r_{rot\_step} = -\lambda_{rot\_step} \exp\left(m_{rot\_step}^2\right)$$

All weight coefficients are listed in Table 4. We train our oracle policy with PPO [5], and the training hyper-parameters are shown in Table 3.

B.2 Training Randomization

To improve the robustness of the oracle policy and diversify the demonstration set, we apply extensive domain randomization during training. We randomize the objects' mass and center of mass, friction coefficients, the shelf's upper-barrier height, camera pose, and the end-effector's initial pose; in addition, we inject noise into observations to emulate real-world imperfections such as calibration bias and readout noise.

Hyper-parameters	Values
Num. envs	*1024*
Num. steps for per update	*24*
Num. minibatches	4
Num. learning epochs	*1500*
learning rate	*0.0003*
clip range	*0.2*
entropy coefficient	*0.0*
kl threshold	*0.02*
max gradient norm	*1.0*
λ	*0.95*
γ	*0.99*

Table 3: Hyper-parameters for the oracle policy learning.

Hyper-parameters	Values
λ_{task}	*5.0*
λ_{action_range}	*1.5*
λ_{action_rate}	*0.0001*
λ_{pose}	*0.6*
$\lambda_{penetration}$	*9.0*
λ_{target_move}	*0.06*
λ_{trans_step}	*10.0*
λ_{rot_step}	*10.0*

Table 4: Hyper-parameters for the reward function.

B.3 Curriculum Learning Methods

We explore various curriculum learning strategies. The first step is to assign difficulty levels to the task scenarios. Specifically, we generate a diverse set of scenarios based on predefined rules, and then categorize them into five difficulty levels according to the occlusion rate—defined as the percentage of the target object's surface area occluded by obstacles when viewed from the front.

Approach 1. ***Per-Environment Difficulty Curriculum.*** ✗ We adopt a locomotion-style curriculum [13] as our first attempt: in Isaac Gym, each environment is assigned a difficulty level; upon task success its difficulty increases, otherwise it decreases. However, this scheme does not yield any improvement over direct training on all scenarios.

Approach 2. ***Unified Difficulty Curriculum.*** ✗ Our second attempt departs from the locomotion-style curriculum. In Isaac Gym, all parallel environments are kept at the same difficulty level, rather than assigning each environment its own difficulty. Once the policy converges at the current difficulty, we advance all environments to the next level. However, this method also fails to produce satisfactory results.

Approach 3. ***Policy-Driven Difficulty Curriculum.*** ✗ Our third attempt abandons the use of manually defined metrics (i.e., occlusion rate) for defining difficulty levels, rather than relying on the policy's own performance. Specifically, we first train the policy on all scenarios until convergence, then evaluate it and label the failed scenarios as higher-difficulty cases. Training then continues exclusively on these failed scenarios, and this process is repeated for five iterations. However, this method also fails to deliver satisfactory results.

Approach 4. ***Task-Conditioned Curriculum.*** ✓ Our final method is a σ-based curriculum (§ 3.3): σ is large early for exploration and is gradually reduced to enhance precision and stability.

Our key insight is that, whether difficulty is defined manually or in a policy-driven way, the chosen scene difficulty axis is highly homogeneous and lacks qualitative transitions across levels; knowledge learned in easy scenarios transfers poorly to hard ones, failing to provide effective learning gradients or a skill ladder. Unlike scene-based curricula, the σ-based curriculum decouples training from scene parameters and schedules solely based on task difficulty via σ, thereby avoiding scene-difficulty specification and directly improving task performance and training stability.

B.4 The Architecture of Oracle Policy

Encoder. The oracle policy receives an observation composed of three parts: the scene representation $\boldsymbol{z}_t$, proprioception $\boldsymbol{p}_t$, and the previous action $\boldsymbol{a}_{t-1}$. The term $\boldsymbol{p}_t \in \mathbb{R}^{12}$ encodes the end-effector pose relative to the target placement and decomposes as $\boldsymbol{p}_t = [\mathrm{rot}_t, \mathrm{trans}_t]$, where

$\mathrm{rot}_t \in \mathbb{R}^9$ denotes the rotation matrix (flattened to 9 entries) and $\mathrm{trans}_t \in \mathbb{R}^3$ is the translation vector. The previous action satisfies $\boldsymbol{a}_{t-1} \in \mathbb{R}^6$. The scene encoder network (§ 3.3) maps the raw scene input to $\boldsymbol{z}_t \in \mathbb{R}^{64}$. The final observation is the concatenation $\boldsymbol{o}_t = [\, \boldsymbol{z}_t,\, \boldsymbol{p}_t,\, \boldsymbol{a}_{t-1} \,] \in \mathbb{R}^{82}$.

Decision. The decision module is a three-layer MLP that maps the observation to an action. Formally, the policy $\boldsymbol{\pi}$ implements $\boldsymbol{\pi} : \mathbb{R}^{82} \to \mathbb{R}^6$ and $\boldsymbol{a}_t = \boldsymbol{\pi}(\boldsymbol{o}_t)$, where $\boldsymbol{a}_t \in \mathbb{R}^6$.

C Vision Policy Training

C.1 Ablation Study on Scaling the Training Data

In our ablation study on scaling the training data, we examine its impact on both the occupancy pre-training (Fig. 11) and the policy learning (Fig. 12). We use suction cup as the end-effector tool to complete this experiment.

Scaling the Training Data for Occupancy Pre-Training. This study examines how the amount of data utilized for pre-training the 3D vision encoder via an occupancy prediction task affects the ultimate performance of the model. All experiments employ varying dataset sizes for pre-training the 3D vision encoder but maintain a fixed dataset size of 500K for policy training. We find a clear correlation between larger datasets and improved policy performance. Starting with 500 scenes, the success rate is 62.33%. As the dataset size increases to 5k and 50k, the success rate improves to 70.72% and 72.50%, respectively. As the dataset size increases to 100k and 250k, the performance continues to improve. The largest dataset, with 500k scenes, achieves the best performance, reaching a success rate of 81.46%. This demonstrates that pre-training on larger datasets significantly enhances the policy performance, providing a more comprehensive understanding of the 3D scene.

Scaling the Training Data for Policy Learning. We seek to determine how the quantity of data available for policy learning influences the model's ultimate performance. For this investigation, all experiments maintain a fixed data volume for 3D vision encoder pre-training but use different amounts of data for training the policy. With a dataset of 500 state-action pairs, the success rate is 48.23%. As the dataset increases to 5k and 50k, the success rate rises to 54.66% and 65.55%, respectively. As the dataset size increases to 100k and 250k, performance continues to improve. The largest dataset of 500k state-action pairs yields the best performance, achieving a success rate of 81.46%. These results highlight that increasing the training data size for the decision module significantly improves task success, emphasizing the importance of a sufficiently large dataset for accurate and reliable policy performance.

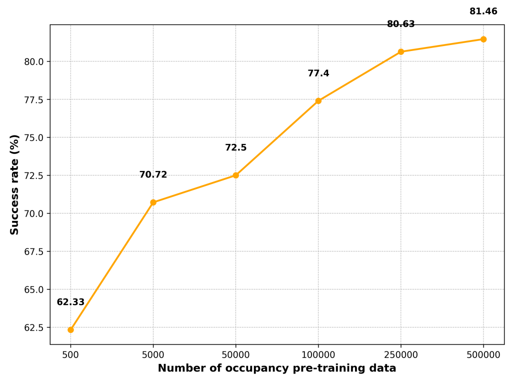


Figure 11: Occ Pre-Training Scaling Law

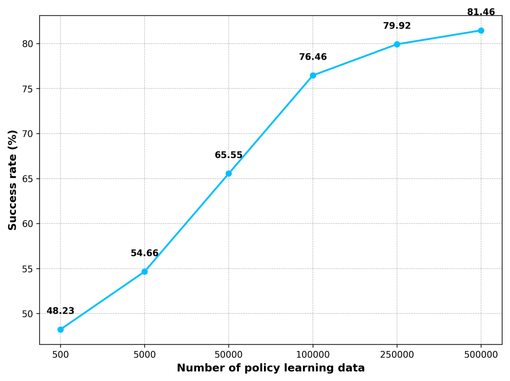


Figure 12: Policy Learning Scaling Law

Overall, our findings highlight the critical role of large-scale data in both occupancy pre-training and policy learning, underscoring the importance of scaling the dataset for improving the overall performance and accuracy of the 3D vision policy.

C.2 Ablation Study on Region of Interest (ROI) Size

Focusing solely on the region of interest is an effective approach that enhances the policy's generalization ability and aids policy learning. We define our Region of Interest (ROI) by cropping an $H \times W \times Z$ cm cubic space around the end-effector.

In our ablation study on the ROI size (Table. 5), we investigated the impact of varying sizes on policy performance. The results show that a compact ROI, specifically $20 \times 20 \times 30$ cm, leads to the best performance, achieving the highest success rate (81.46%) and the lowest translation (2.78 cm) and rotation (0.36 rad) errors. It is crucial to note that ***the optimal ROI size is not arbitrarily small but is closely tied to the scale of the target objects.*** In our experiments, the target objects average approximately $5 \times 10 \times 12$ cm. Therefore, an ROI of $20 \times 20 \times 30$ cm is large enough to fully encompass the target while providing essential local context, yet small enough to filter out most distracting information. As the ROI size increases beyond this optimal range, both the success rate and accuracy metrics decrease. This is because larger ROIs introduce more irrelevant information from distant areas, which reduces the model's ability to focus on the target region and leads to less precise predictions.

These findings highlight the advantage of using a ***task-appropriate, localized ROI.*** This approach not only improves policy performance but also enhances generalization, making the network more robust to scene variations. It also accelerates policy inference while maintaining high performance, especially in cluttered environments.

ROI Size (cm)	Success Rate (%) ↑	Translation (cm)↓	Rotation (rad)↓
$20 \times 20 \times 30$	**81.46%**	**2.78**	**0.36**
$40 \times 80 \times 30$	76.93%	3.24	0.43
$60 \times 150 \times 30$	74.63%	3.67	0.47

Table 5: **Ablation study on Region of Interest (ROI) size.** A compact ROI of $20 \times 20 \times 30$ cm achieves the best results, with the highest success rate and lowest errors. This highlights the benefit of a localized ROI that is appropriately scaled to the task, as larger ROIs degrade performance by including irrelevant information from distant areas.

C.3 Training Details

Pre-training for occupancy prediction used 40 GeForce RTX 4090 GPUs on 500K scenes and finished in 16 hours. Policy learning then trained on 500K demonstration frames for 8 hours. In real-world tests, the system runs at over 10 fps on a single NVIDIA GeForce RTX 4090.

C.4 The Details of 2D-3D Spatial Attention

Rather than averaging multi-view features (which assumes equal contribution), we fuse them using Deformable Cross-Attention (DCA) [12], guided by a grid of 3D queries $Q \in \mathbb{R}^{C \times H \times W \times Z}$.

As shown in Fig. 7, for each 3D query, defined by its position p and feature q_p, we project it onto each camera view t via $\mathcal{P}(p,t) : \mathbb{R}^3 \mapsto \mathbb{R}^2$. We retain only the set of views where the projection is valid, $\mathcal{V}_{\text{hit}}$. The aggregated 3D feature $\mathcal{F}^p$ at voxel p is then computed by averaging the attention outputs from all visible views:

$$\mathcal{F}^p = \text{DCA}(q_p, \{\mathcal{F}_t^{2D}\}) = \frac{1}{|\mathcal{V}_{\text{hit}}|} \sum_{t \in \mathcal{V}_{\text{hit}}} \text{DA}\big(q_p, \mathcal{P}(p,t), \mathcal{F}_t^{2D}\big),$$

where $\mathcal{F}_t^{2D}$ is the 2D feature map of view t. The Deformable Attention (DA) module is defined as:

$$\text{DA}(q, p_{\text{ref}}, X) = \sum_{m=1}^{N_{\text{head}}} W_m \left(\sum_{k=1}^{N_{\text{key}}} A_{mk} \cdot W'_m X(p_{\text{ref}} + \Delta p_{mk}) \right).$$

Here, for each attention head m and key k, the module uses the query q to predict a 2D sampling offset Δp_{mk} relative to the reference point $p_{\text{ref}} = \mathcal{P}(p,t)$ and an attention weight A_{mk}. It then samples the 2D feature map X at these sparse, dynamically determined locations ($p_{\text{ref}} + \Delta p_{mk}$) and computes a weighted sum. W_m and W'_m are standard learnable projection matrices.

This approach allows the final 3D feature volume, $\mathcal{F} = \{\mathcal{F}^p\}$, to unequally weight views based on visibility and content, enhancing robustness to occlusion and blur.

C.5 Processing of Relative Depth

Depth estimates from DepthAnything [9] are inherently scale-ambiguous. To create a consistent input for our feature extractor, we first normalize each predicted depth map using a per-image **min-max scaling** to bring its values into the range $[0, 1]$.

Concretely, given an RGB image $I \in \mathbb{R}^{H\times W\times 3}$, DepthAnything produces a single-channel depth map $D \in \mathbb{R}^{H\times W}$. This map is normalized as:

$$\hat{D} = \frac{D - \min(D)}{\max(D) - \min(D) + \varepsilon}$$

where $\min(D)$ and $\max(D)$ are the minimum and maximum values of the specific depth map D, and ε is a small constant for numerical stability.

To prepare this normalized map for a standard ResNet backbone, we treat $\hat{D}$ as a grayscale image. It is first replicated across three channels to match the expected input format ($H \times W \times 3$). Then, this 3-channel tensor undergoes a second normalization using the standard ImageNet mean and standard deviation values. The resulting tensor is finally fed into the ResNet backbone (with the final max-pooling layer removed) to extract the feature map F.

C.5.1 Mapping Relative Depth to Metric Voxels

Our method does not perform an explicit conversion from relative depth to metric depth, nor does it use depth values to unproject 2D features into a 3D point cloud. Instead, the core of our approach is to project the centers of a predefined 3D metric voxel grid (with a cell size of 5 mm) onto the input camera views. This forward projection is based on the standard pinhole camera model and utilizes the provided camera matrices.

This multi-view projection strategy is key to resolving spatial ambiguity. By using multiple views, points that might lie along a single viewing ray from one camera are disambiguated through parallax, as they project to distinct 2D locations in the second view.

Consequently, the network is tasked with implicitly learning the transformation from relative depth features to a metrically-scaled occupancy grid, guided by these strong multi-view geometric constraints. The entire process is optimized end-to-end with direct supervision from the ground-truth metric voxel data. In summary, our methodology leverages the geometric accuracy of the pinhole camera model for projection and the power of end-to-end learning to fuse features from relative depth into a usable metric representation.

C.6 Real-World Reconstruction Results

We first pre-train the encoder on an occupancy-prediction auxiliary task in simulation, which yields high-quality reconstruction not only in simulation but also on real hardware (strong sim-to-real, as shown in Fig. 13). Under heavy occlusions, the model infers the occluded volume to recover a complete scene representation, and it remains robust to transparent, reflective, and irregular objects. Importantly, we do not feed the reconstructed voxels (final occupancy map) to the downstream policy. Instead, the policy receives intermediate latent features from the pre-trained encoder; although it never observes the final occupancy prediction, these auxiliary-task–enriched features allow it to implicitly capture the scene's complete 3D geometry.

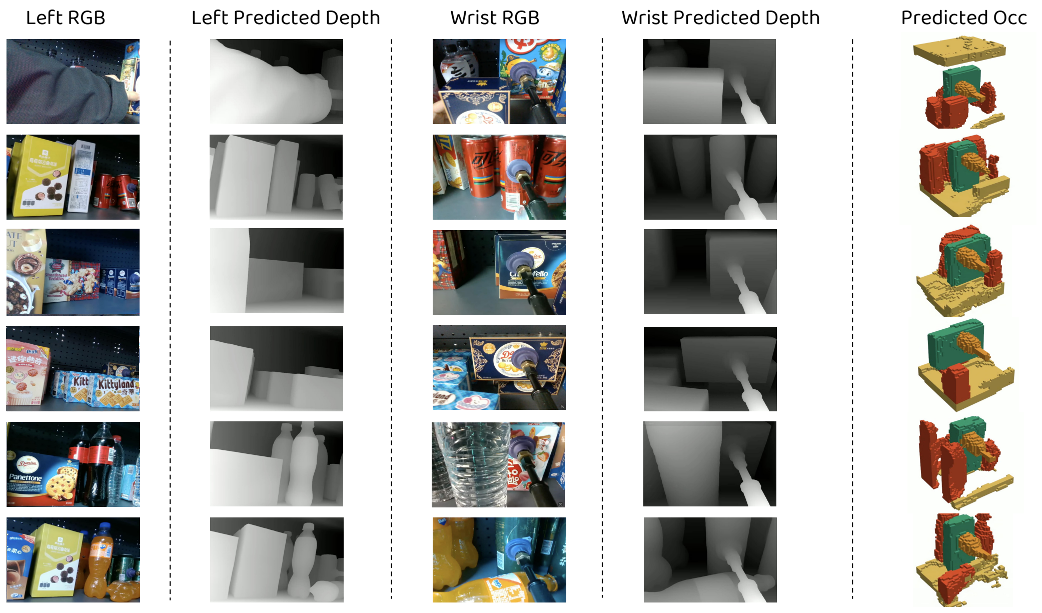
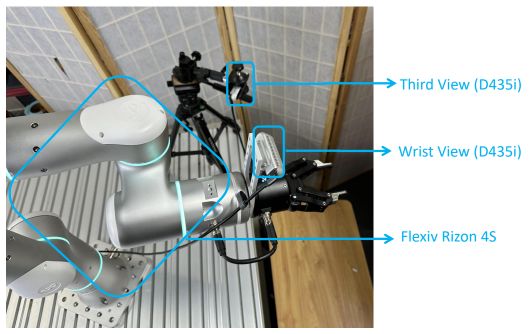
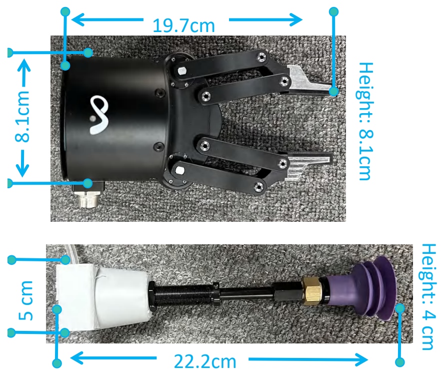


Figure 13: Real-world occupancy reconstruction results, capable of handling diverse scenarios: varying object shapes, different materials, and complex layouts.



(a) Hardware Setup



(b) End-Effector Tools

Figure 14: We illustrate our real-robot experimental hardware setup and the two types of end-effector tools employed: one suction cup and one parallel gripper.

D Baseline Implementation Details

Baselines utilizing RGB (Diffusion Policy), point cloud (3D Diffusion Policy), RGB-D voxels, raw depth, and predicted depth all share the same action generation method, differing only in their input representations. Actions are generated via a Denoising Diffusion Probabilistic Model (DDPM) [2], which utilizes 1000 denoising steps during training and 100 steps for inference, its network architecture is a three layer MLP.

RGB (Diffusion Policy) [1]. We use dual-view RGB images as input and extract features following the same processing pipeline as Diffusion Policy (i.e., through ResNet and Spatial Softmax). Subsequently, actions are generated using DDPM.

Point Cloud (3D Diffusion Policy) [11]. Our input is a point cloud generated by fusing two camera depth maps. Following the approach in DP3, we employ a Simple PointNet to extract features from this point cloud.

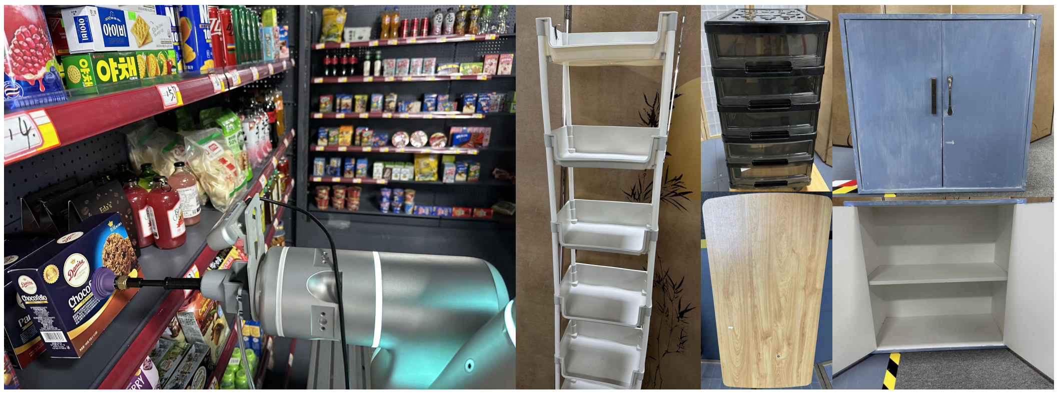


Figure 15: Illustration of experimental scenarios. The main experiments are conducted in a replicated retail store, supplemented by evaluations on a storage rack, in a cabinet, in a drawer, and on tabletops.

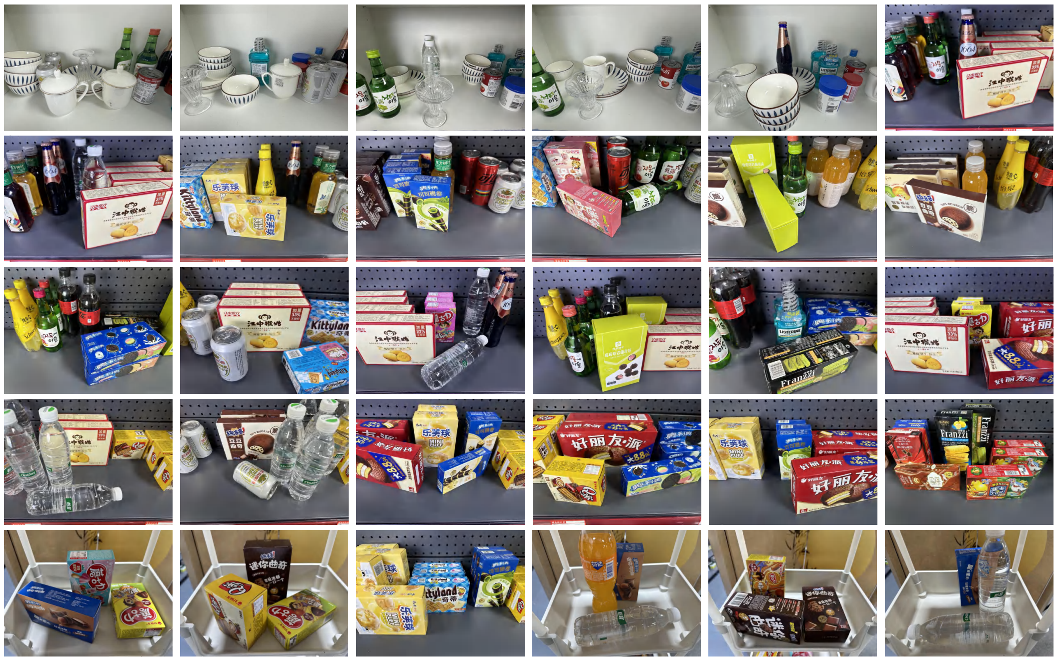


Figure 16: Layouts for real-robot experiments.

RGB-D Voxel. We project RGB images from two viewpoints into 3D space using camera extrinsics and depth maps, thereby forming a voxel representation. A 3D convolutional network (3D ConvNet) is then employed to extract features from these voxels. Then use DDPM to generate actions.

Raw Depth. We use ResNet to extract features from two viewpoint depth maps, after which actions are generated via DDPM.

Predicted Depth. We utilize Depth Anything to obtain predicted depth maps. Features are then extracted from these maps using ResNet. Subsequently, actions are generated via DDPM.

E Real-Wold Experiment Details

E.1 Setup

Hardware. In our real-world experiments, we use the Flexiv Rizon 4S robotic arm. Two Intel D435i cameras provide input for our 3D vision policy from different perspectives, exclusively utilizing their

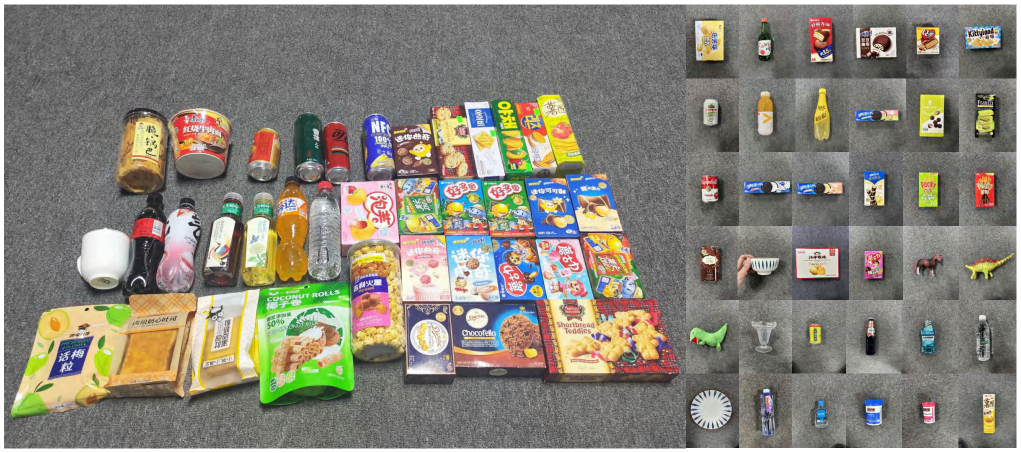


Figure 17: Objects used in real-robot experiments, including a diverse range of items with various shapes (such as boxed, bottled, and irregular forms) and materials (including transparent and reflective types).

RGB streams. One camera is mounted on the robotic arm, while the other offers a third-person view, as shown in Fig. 14 (a).

End-Effector Tools. We utilize two end-effector tools: a suction cup and a parallel gripper. The suction cup, which was custom-made by us via 3D printing, is used to handle boxed objects. The parallel gripper, Flexiv GRAV model, is employed for grasping bottled objects (illustrated in Fig. 14 (b)).

Experiment Scenes. Our main real-world experiments are primarily conducted in a replicated retail environment. Additionally, we perform experiments in settings involving a cabinet and a rack, all scenes all shown in Fig. 15. And all 30 experimental layouts are shown in Fig. 16.

E.2 Real-World System Design

Our real-world system employs a ***hierarchical, asynchronous control architecture*** to ensure smooth and stable robot execution. The system is decoupled into two main layers: a high-level ***Policy Layer*** and a low-level ***Control Layer***.

The ***Policy Layer*** operates at a low frequency of ***10 Hz***. In this layer, the policy network receives the latest sensor observations, performs inference, and generates a target end-effector (EE) pose (position and orientation). This target pose is then sent as a command to the lower-level controller.

The ***Control Layer*** is an ***IK-based controller*** that runs asynchronously at a high frequency of ***100 Hz***. This layer is responsible for translating the target EE pose into a smooth trajectory of joint positions (`qpos`). It receives commands from the policy layer asynchronously and interpolates the robot's current pose towards the target. In each control cycle, it calculates a small, kinematically valid motion step using an inverse kinematics solver, ensuring that the robot's velocity limits are respected. This decoupling of low-frequency policy decisions from high-frequency motion control is crucial for stable and continuous real-world performance.

E.3 Experiment objects

The objects used in our experiments encompass a diverse range of types, including those that are boxed, bottled, transparent, reflective, or irregularly shaped. The experimental objects are displayed in Fig. 17.

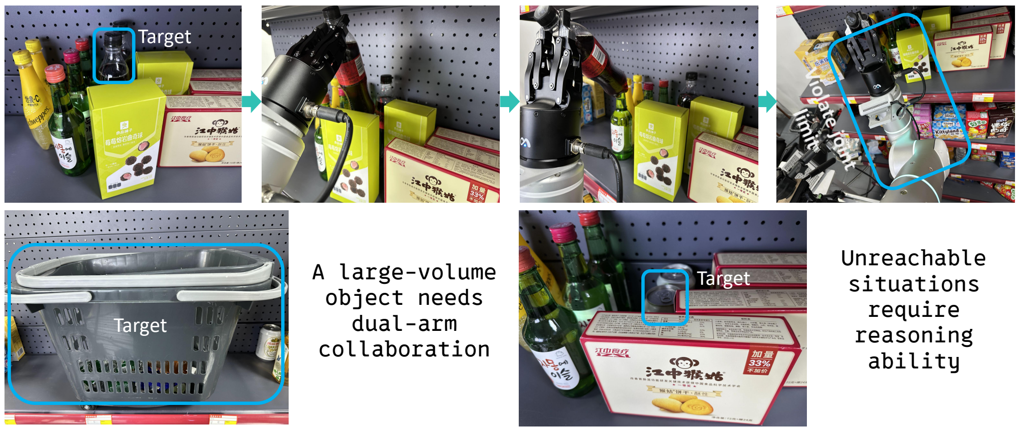

Target
Violate joint limit
Target
A large-volume object needs dual-arm collaboration
Target
Unreachable situations require reasoning ability

Figure 18: Illustration of Limitations. (Top row) Actions exceeding joint limits due to an attempted approach from above. (Bottom row, left) A large-volume object demonstrating the need for dual-arm manipulation. (Bottom row, right) A completely occluded and initially unreachable target, necessitating a multi-step reasoning process involving moving the obstacle, fetching the target, and subsequently returning the obstacle to its original position.

F Failure Modes Analysis

When scenes become excessively complex, there is a possibility that the policy's output will lead to joint limit violations for the robotic arm, causing the task to fail. The top of Fig. 18 depicts a scenario where the robot's intended rotational approach from above to fetch an object results in exceeding its joint limits and a collision with the upper barrier. Furthermore, when an object is too large and too heavy, we need to employ dual-arm coordination, as shown in the bottom of Fig. 18. Finally, when the target is completely occluded to the point of being unreachable, reasoning capabilities are required to first move the obstacles aside, then retrieve the target, and subsequently restore the obstacles to their original positions (Fig. 18). All of these can be considered as future work.

References

[1] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[3] Yitong Li, Ruihai Wu, Haoran Lu, Chuanruo Ning, Yan Shen, Guanqi Zhan, and Hao Dong. Broadcasting support relations recursively from local dynamics for object retrieval in clutters. In *Robotics: Science and Systems*, 2024.

[4] Yaoyao Qian, Xupeng Zhu, Ondrej Biza, Shuo Jiang, Linfeng Zhao, Haojie Huang, Yu Qi, and Robert Platt. Thinkgrasp: A vision-language system for strategic part grasping in clutter. *arXiv preprint arXiv:2407.11298*, 2024.

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[6] Kentaro Wada, Stephen James, and Andrew J Davison. Safepicking: Learning safe object extraction via object-level mapping. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10202–10208. IEEE, 2022.

[7] Yongliang Wang and Hamidreza Kasaei. Learning dual-arm push and grasp synergy in dense clutter. *arXiv preprint arXiv:2412.04052*, 2024.

[8] Kechun Xu, Shuqi Zhao, Zhongxiang Zhou, Zizhang Li, Huaijin Pi, Yifeng Zhu, Yue Wang, and Rong Xiong. A joint modeling of vision-language-action for target-oriented grasping in clutter. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11597–11604. IEEE, 2023.

[9] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv preprint arXiv:2406.09414*, 2024.

[10] Yuxiang Yang, Jiangtao Guo, Zilong Li, Zhiwei He, and Jing Zhang. Ground4act: Leveraging visual-language model for collaborative pushing and grasping in clutter. *Image and Vision Computing*, 151:105280, 2024.

[11] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *ICRA 2024 Workshop on 3D Visual Representations for Robot Manipulation*, 2024.

[12] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

[13] Ziwen Zhuang, Zipeng Fu, Jianren Wang, Christopher Atkeson, Sören Schwertfeger, Chelsea Finn, and Hang Zhao. Robot parkour learning. In *Conference on Robot Learning (CoRL)*, 2023.