AnyPlace: Learning Generalizable Object Placement for Robot Manipulation

Yuchi Zhao[1, 2], **Miroslav Bogdanovic**[1, 2], **Chengyuan Luo**[3], **Steven Tohme**[4], **Kourosh Darvish**[1, 5], **Alán Aspuru-Guzik**[1,2,5], **Florian Shkurti**[1,2], **Animesh Garg**[6]
[1]University of Toronto, [2]Vector Institute, [3]Shanghai Jiao Tong University, [4]Wilfrid Laurier University, [5]Acceleration Consortium, [6]Georgia Institute of Technology

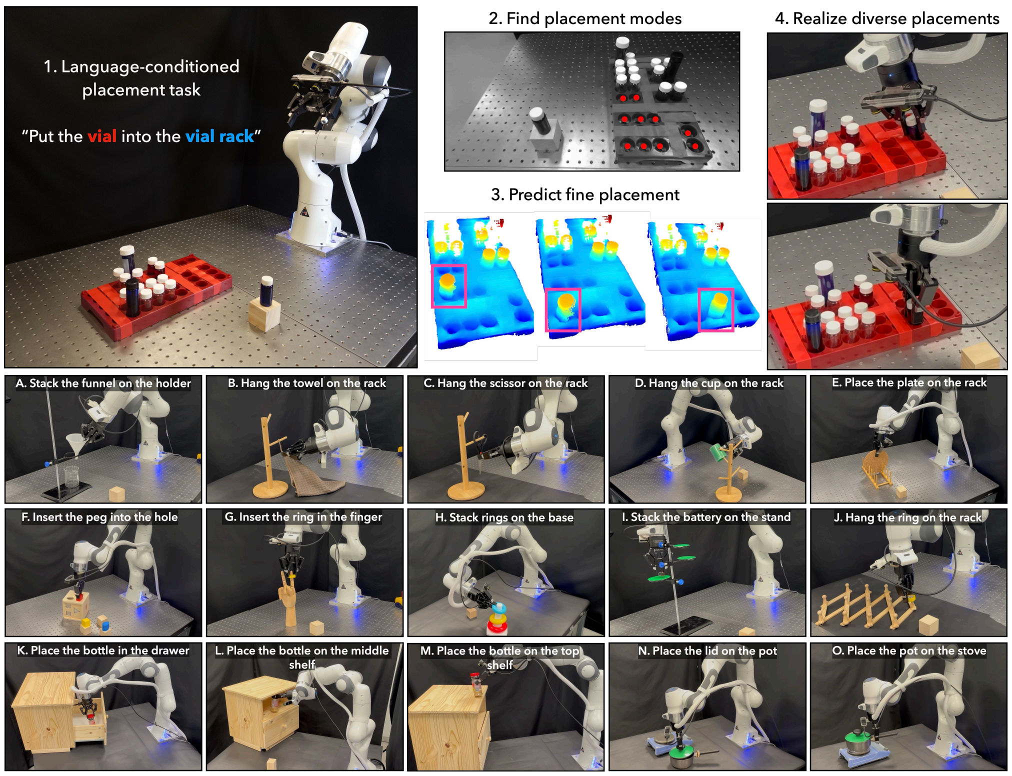

1. Language-conditioned placement task
"Put the vial into the vial rack"
2. Find placement modes
3. Predict fine placement
4. Realize diverse placements
A. Stack the funnel on the holder
B. Hang the towel on the rack
C. Hang the scissor on the rack
D. Hang the cup on the rack
E. Place the plate on the rack
F. Insert the peg into the hole
G. Insert the ring in the finger
H. Stack rings on the base
I. Stack the battery on the stand
J. Hang the ring on the rack
K. Place the bottle in the drawer
L. Place the bottle on the middle shelf
M. Place the bottle on the top shelf
N. Place the lid on the pot
O. Place the pot on the stove

Figure 1: Execution of the AnyPlace approach by the robot. (1) Given a language description of a placement task, the robot first captures an RGBD image of the scene using its eye-in-hand camera. (2) A segmentation model and a VLM are used to segment objects and suggest possible placement locations. (3) Multiple placement poses are predicted for objects around suggested locations. (4) The robot realizes placement into any of the predicted poses. (A-O) AnyPlace shows generalization and robustness in predicting placement poses across 16 tasks in the real world, despite being trained purely on a small synthetic dataset.

Abstract: Object placement in robotic tasks is inherently challenging due to the diversity of object geometries and placement configurations. We address this with AnyPlace, a two-stage method trained entirely on synthetic data, capable of predicting a wide range of feasible placement poses for real-world tasks. Our key insight is that by leveraging a Vision-Language Model (VLM) to identify approximate placement locations, we can focus only on the relevant regions for precise local placement, which enables us to train the low-level placement-pose-prediction model to capture multimodal placements efficiently. For training, we generate a fully synthetic dataset comprising 13 categories of randomly generated objects in 5370 different placement poses across three configurations (insertion, stacking, hanging) and train local placement-prediction models. We extensively evaluate our method in high-fidelity simulation and show that it consistently outperforms baseline approaches across all three tasks in terms of success rate, coverage of placement modes, and precision. In real-world experiments, our method achieves an average success and coverage rate of 76% across three tasks, where most baseline methods fail completely. We further validate the generalization of our approach on 16 real-world placement tasks, demonstrating that models trained purely on synthetic data can be directly transferred to the real world in a zero-shot setting. More at: any-place.github.io.

Keywords: Pick and Place, Robot Manipulation, Synthetic Dataset

1 Introduction

Placing objects is a fundamental task that humans perform effortlessly in daily life, from setting items on a table to inserting cables into sockets. On the other hand, enabling a robot to perform such tasks can often be highly challenging. The challenges arise from the various constraints of different placement tasks and the difficulty of generalizing to unseen objects. Additionally, predicting multimodal placement outputs, which encompass a range of valid locations and modes, remains challenging, particularly when multiple feasible solutions exist. Existing methods are often task-specific, using *a large number of demonstrations for a single placement task*, such as hanging objects on racks [1], hoping that the robot can generalize to unseen objects. Alternatively, few-shot approaches [2, 3, 4, 5, 6, 7, 8, 9, 10, 11] focus on learning object placement with a few demonstrations, aiming for the model to *replicate the same placement operation* across random initial configurations of similar objects and setups. However, both struggle with generalization and scalability.

We reframe the placement problem as a *pairwise shape mating* [12], allowing us to *unify multiple placement tasks into a single learning objective*. In this work, we address generalizable object placement that is robust to different objects and capable of predicting diverse and precise placement poses across various tasks, such as placing in open areas, inserting in fine slots, and hanging. We have developed a fully synthetic dataset that captures three common placement configurations: *inserting, stacking, and hanging*. Furthermore, we develop a placement prediction algorithm that consists of a high-level placement position proposal module and a low-level placement pose prediction model. We use a spatial VLM to propose all possible placement locations and extract the local point cloud regions based on them. By providing a coarse region for the low-level module to focus on, the model can effectively generalize across objects, learn their geometry, and capture diverse placement configurations. We use a diffusion model for fine pose prediction, conditioned on a small local point cloud region, which enables precise and multimodal placement pose predictions. We demonstrate the effectiveness of our methods across a range of placement tasks in simulation, as well as diverse real-world tasks, outperforming the baseline models in terms of success rate and coverage in both cases. The key contributions of our work are:

1. We propose a novel object placement approach that leverages a VLM to reason about potential placement locations and a low-level pose prediction model to predict placement poses based solely on the region of interest. We show that this coarse-to-fine mechanism allows us to significantly improve performance with respect to baseline methods in terms of success rate, precision, and placement mode coverage.
2. We develop a data generation pipeline and build a fully synthetic dataset containing thousands of generated objects and capturing a wide range of local placement configurations.
3. We demonstrate our approach generalizes to the real world on 16 placement tasks with novel objects and varying degrees of precision requirements.

2 Related Work

The problem of robot pick-and-place is typically formulated in two ways: object rearrangement and direct end-effector pose prediction.

Object Rearrangement In object rearrangement, the goal is to train a model to predict the relative transformation of the object from its initial pose to its final placement pose. In this setting, many of the works focus on predicting explicit task-relevant features of both objects and then solving for the relative pose through optimization or regression. Specifically, the Neural Descriptor Fields (NDF) series of papers [2, 3, 4] learn the occupancy field of point clouds as a representation. A fixed set of keypoints on the placement object queries features from the target's field, and the best transformation is estimated by matching these to demonstration features. TaxPose [5] leverages transformer-based cross-attention to predict corresponding points between two objects and uses differentiable singular value decomposition (SVD) to solve for the relative transformation. To guarantee the placement pose prediction model is robust to SE(3) transformations, i.e., SE(3)-equivariant, methods [6, 7, 8] explicitly predict per point type-0 and type-1 features for object point clouds and then solve an optimization problem to align these features into specific configurations based on demonstrations. All of these methods operate in a few-shot setting and can predict a single placement pose given two objects. It is crucial for models to capture and predict a distribution of placement poses, as not every placement pose is realizable by a robot due to its kinematic constraints. RPDiff [1], by contrast, trains a transformer with a diffusion mechanism on a large dataset, gradually denoising the object placement pose. However, their experiments reveal that the coverage of possible placement locations is incomplete. The fixed-size cropping mechanism used during diffusion may also struggle to generalize to objects of varying sizes. Additionally, a recent study [13] samples multiple stable placements in a simulation and employs a VLM to select the appropriate mode based on a language query. While these modes are discrete, each mode allows for the rotation of objects along their axis of symmetry, resulting in valid placement poses that form a continuous distribution.

Direct Pick and Place End-effector Pose Prediction An alternative approach to the pick-and-place task is predicting the robot's end-effector pose directly. M2T2 [14] and Pick2Place [15] focus on planar object placement in cluttered scenes. M2T2 [14] employs a multi-task transformer with separate decoders to predict grasp poses and placement location affordance maps for each discrete bin of rotation. Other works, like Pick2Place [15], concentrate on predicting key end-effector poses to accomplish specific tasks. RVT [16] and RVT-2 [17] also utilize a transformer, leveraging multiview RGB images of the scene to predict heatmaps for the robot's next end-effector location. Coarse-to-fine Q-attention [18], on the other hand, leverages the scene's voxel to identify the most interesting spatial point at the current resolution. This point becomes the voxel centroid for the next refinement step, enabling the model to gather more accurate 3D information. Another line of work on contemporary VLMs and vision foundation models (VFMs) [19] finds that they often lack reliable spatial reasoning abilities, limiting their effectiveness in fine-grained manipulation tasks such as peg-in-hole insertion, where precise object placement is critical.

3 AnyPlace: Generalizable Object Placement

To enable a robot to execute diverse object placements in a scene, we propose decomposing the placement pose prediction problem into two stages: a high-level coarse placement location proposal stage and a low-level fine placement pose prediction stage. For the high-level task, we incorporate a vision-language model, trained to output 2D keypoint locations in an image based on a given text prompt. A small local region around the candidate placement location can then be extracted for the low-level pose-prediction model. This simplifies the low-level pose prediction problem significantly by using a much smaller point cloud as input and improves generalization overall, as *features outside the local region do not influence the prediction*. This allows us to focus on a limited set of general placement types and utilize a fully synthetic dataset, but have the final model be effective in a broad range of real-world placement tasks. Additionally, the high-level prediction stage enables the identification of multiple placement modes.

Problem setup. We formulate the object placement task as predicting relative transformations. Specifically, given an input tuple $\{D, I\}$, where D represents the language description of the placement task and I is an RGBD image of the scene, our goal is to predict a set of rigid transformations $\{T_n\}_{n=1}^{N} \subset \mathrm{SE}(3)$ that move the target object C from its current position to all viable placement locations on the base object B that satisfy language conditioning D. Assuming the grasping poses T_{grasp} are provided by a grasp prediction model, the final end-effector pose T_{place} can be computed

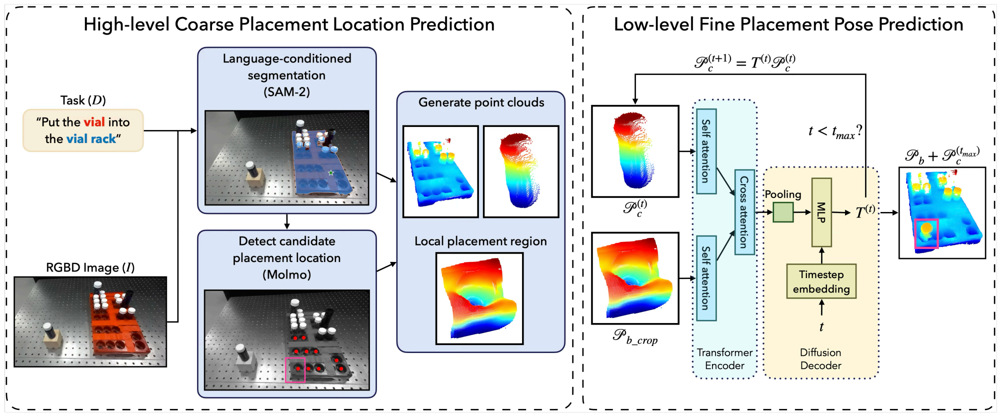

High-level Coarse Placement Location Prediction
Task (D)
"Put the vial into the vial rack"
RGBD Image (I)
Language-conditioned segmentation (SAM-2)
Detect candidate placement location (Molmo)
Generate point clouds
Local placement region
Low-level Fine Placement Pose Prediction
$\mathscr{P}_c^{(t+1)} = T^{(t)}\mathscr{P}_c^{(t)}$
$\mathscr{P}_c^{(t)}$
$\mathscr{P}_{b\_crop}$
Self attention
Self attention
Cross attention
Transformer Encoder
Pooling
MLP
Timestep embedding
t
Diffusion Decoder
$T^{(t)}$
$t < t_{max}$?
$\mathscr{P}_b + \mathscr{P}_c^{(t_{max})}$

Figure 2: Overview of the AnyPlace placement pose prediction approach. (1) High-level Coarse Placement Location Prediction: Given an input language description and an RGBD image, we leverage a VLM and a segmentation model to extract objects of interest. We then prompt the VLM to propose possible placement locations and crop the region of interest centered on the proposed points. The resulting point clouds are fed into the low-level model. (2) Low-level Fine Placement Pose Prediction: We use a transformer architecture, with a diffusion decoder to predict relative transformation of the point clouds to realize placement.

using the predicted relative transformation between the initial object pose and its final placement pose as $T_{\text{place}} = T_n T_{\text{pick}}$.

3.1 VLM-guided coarse placement location prediction

When multiple potential placement locations exist within a task, existing models often struggle to capture all possibilities. To address this, we propose leveraging spatial VLMs, which have demonstrated strong capabilities in localizing points and regions within images based on language descriptions, to directly identify placement locations. Specifically, given a language description of the placement task D and a RGBD image I, we extract the point cloud of the target object $\mathcal{P}_{\text{c}}$, and for each identified placement point in the image, we extract the local region of the base object $\mathcal{P}_{\text{b\_crop}}$ guided by the 3D bounding box of the target object, where $\mathcal{P}_{\text{c}}, \mathcal{P}_{\text{b\_crop}} \in \mathbb{R}^{N\times 3}$. These point clouds are then used as input to the pose prediction model.

This approach enables the low-level pose prediction model to focus on learning different placement configurations of two objects and predicting placement poses. Also, explicitly identifying placement modes, rather than relying on models to explore placements across the entire object, is more reliable and practical when handling diverse objects with multiple possible placement poses. Since our high-level module is built on a general-purpose VLM, the system can also handle diverse placements and perform complex language conditioning. We utilize Molmo [20] to detect all potential placement locations as keypoints in image space, such as specifying all positions where a vial can be inserted into a vial plate. However, our approach is not specific to one VLM, and other models with capabilities to give point locations in an image could be used [21]. (See Appendix A for language prompts.)

3.2 Fine-grained placement pose prediction

Given point clouds $\mathcal{P}_{\text{c}}$ and $\mathcal{P}_{\text{b\_crop}}$ from the high-level module, the low-level pose prediction only focuses on learning different local placement arrangements, without the need to capture the distribution of different discrete placement locations. Our intuition is that, with the aid of our large synthetic dataset, the model should effectively capture key representations of diverse placement configurations based on object geometry, which enables it to generalize to unseen objects and remain robust to noisy data. Having only a local region as input, the pose prediction model should be able to achieve better precision, which is crucial in many relevant placement tasks.

We predict the relative transformation using a diffusion model, which takes as input the two point clouds, $\mathcal{P}_{\text{c}}$ and $\mathcal{P}_{\text{b\_crop}}$, and through iterative denoising produces the transformation to be applied to $\mathcal{P}_{\text{c}}$ in order for it to be placed correctly in the $\mathcal{P}_{\text{b\_crop}}$ region. We use a transformer architecture for the encoder [22, 12], where the features of the two point clouds are first extracted through self-attention layers, before being combined through cross-attention, and pooled into a latent embedding. We use a diffusion architecture for the decoder, which, conditioned on this latent embedding and the diffusion timestep, produces the delta to be applied to the object pose through transformation $T_n^{(t)}$. For each successive diffusion step, we apply this transformation to the original point cloud $\mathcal{P}_c$, in effect shifting the object being placed closer to the goal. The resulting transformation to go from the original point cloud, where the object currently is, to the final placement pose is the product of the output from each diffusion step $T_n = \prod_{t=1}^{t_{\max}} T_n^{(t)}$. At each diffusion step during training, given the ground truth $T_{n,GT}^{(t)}$ and the predicted $T_n^{(t)}$, we use the L1 distance for the translation loss, the geodesic distance for the rotation loss, and apply the Chamfer loss between point clouds transformed by predicted and ground truth poses. The total loss is the sum of these individual losses. (More details in Appendix B.)

Robot pick and place execution After determining the placement poses, we implement a pick-and-place pipeline to manipulate the object and position it accurately at the target pose. Specifically, we utilize AnyGrasp [23] to find viable grasps for the target object C and employ cuRobo [24] as the motion planner to perform collision-free placement. We perform rejection sampling on $(T_{\text{place}}, T_{\text{pick}})$ pairs to identify valid grasps for the specific placement pose predicted by our model that can be executed by the robot. Details of implementation can be found in Appendix C.

4 Synthetic Dataset Generation

Our aim in building the synthetic dataset is to capture a broad range of local placement arrangements. Existing models use a few very specific tasks (e.g., inserting a book into a bookshelf) to simply evaluate how well their model works given placement data for such task for training [14, 15, 25]. Our goal, on the other hand, is not only to use the dataset to evaluate the proposed model, but to build towards representing a broad range of types of placements (stacking, hanging, inserting) as shown in Figure 3. The local nature of our pose-prediction model makes this task much easier and enables us to build a dataset that can generalize to a broad range of real-world placement tasks.

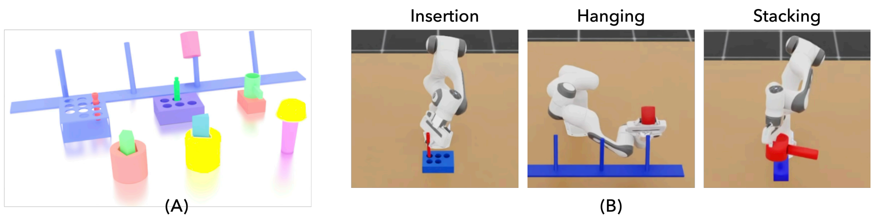


Figure 3: Dataset generation and robot performing various placement tasks in simulation.

The data generation pipeline consists of two main components: object generation and placement pose generation. Specifically, we use Blender to procedurally generate 3D objects with random object parameters to further enhance diversity. To identify stable placements for objects, we use NVIDIA IsaacSim to determine object placement poses for three configurations: stacking, inserting, and hanging. This dataset covers a wide range of placement scenarios encountered in real life. In total, 1489 objects across 13 categories were created, and 5370 placement poses were generated. (For more details, see Appendix B.3.)

5 Experimental Evaluation

We conduct evaluations against three baseline models on different placement tasks in both simulation and real-world settings. We aim to answer the following questions:

1. How well does AnyPlace perform in terms of object placement success rate, coverage, and precision compared to baseline models?
2. How does each component of AnyPlace described in section 3 affect the overall performance?
3. How well can AnyPlace generalize to novel objects and unseen configurations in the real world in a zero-shot setting, despite being trained only on synthetic data?

Evaluation metrics. We use three metrics to evaluate model performance: success rate, coverage, and precision. Specifically, we define a placement as successful if the robot places the object at the

Table 1: Success rate (%) on synthetic dataset. Overall, AnyPlace achieves consistently high success rates across four pick-and-place tasks on the synthetic dataset.

	Methods	Object Stacking (single-mode)	Peg Insertion (single-mode)	Cup Hang (multi-mode)	Vial Insertion (multi-mode)
Single task training	NSM [12]	76.57	7.63	35.54	18.70
	RPDiff [1]	**80.34**	22.94	92.02	16.51
	AnyPlace-EBM (ours)	**80.04**	8.44	91.57	65.64
	AnyPlace (ours)	**80.16**	**30.95**	**94.80**	**92.74**
Multi-task training	NSM [12]	77.55	7.69	35.22	9.87
	RPDiff [1]	**80.21**	22.33	**94.05**	24.26
	AnyPlace-EBM (ours)	78.95	10.75	90.87	57.24
	AnyPlace (ours)	78.28	**24.99**	**94.12**	**75.25**

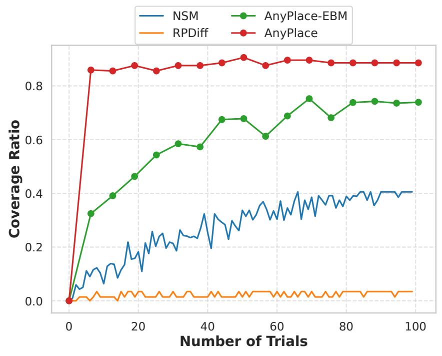
correct location, and it remains stable after release. The success rate is then calculated as the number of successful placements divided by the total number of trials. To better understand the diversity of multimodal outputs in placement prediction, we evaluate coverage, defined as the number of distinct predicted placement locations relative to the total number of possible locations. Finally, for fine placement tasks, to evaluate the precision, we measure the error between the ground truth pose and the predicted pose in terms of both distance and angle.

Baselines. We compare our approach with three baselines: NSM [12], RPDiff [1], and an energy-based model (EBM), a variant of our model that is integrated with our high-level placement location prediction module. Specifically, NSM shares the same self-attention and cross-attention point cloud encoder, paired with a regression decoder. For RPDiff, since the low-level pose-prediction module in our diffusion-based approach shares the same structure, this allows us to directly examine the effects of the high-level module we propose. To evaluate the effectiveness of the diffusion decoder in generating multimodal outputs, we build AnyPlace-EBM inspired by Implicit-PDF [26], where the model uses the same encoders as ours and is trained to assign low energy values to stable placement poses (Details in Appendix C.2). Each model is trained independently on placement-type-specific subsets and on the full dataset (the multitask variant).

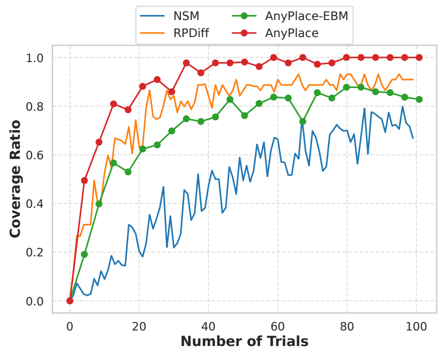
Placement Success: Single & Multi-Modal. We evaluate all methods in simulation using our pick-and-place execution pipeline within IsaacLab [27]. Experiments cover four tasks: `object stacking` and `peg insertion` (single-mode), and `cup hang` and `vial insertion` (multi-modal, with multiple valid placements). A summary of results is shown in Table 1.

In single-mode tasks, for the simple stacking task, where precision in placement poses is not required, all models achieve a similar success rate. For the peg-in-hole task, which requires high precision in placement pose prediction, AnyPlace surpasses the baseline models by a large margin. Multimodal tasks are where we expect to get the full benefit from our approach. In the hanging task, the tolerance for placing a cup on the rack is relatively high, and AnyPlace outperforms baseline models slightly. In the more challenging vial insertion task, AnyPlace achieves the highest success rate of 92.74%, while the success rates of all baseline models drop significantly. This demonstrates that relying on high-level VLM to propose possible placement locations and focusing solely on the local region for placement prediction simplifies the task for low-level pose prediction models and enables them to better capture fine-grained point cloud features for high-precision placements. Additionally, despite using the same high-level placement location prediction, the energy-based model suffers a worse performance compared to the diffusion-based AnyPlace. This highlights that the iterative denoising procedure in diffusion is more effective for high-precision placement prediction.

Placement Coverage: Multi-Modal Insertion & Hanging. We now investigate how coverage changes as the number of samples taken from each model increases. Given the design of NSM and RPDiff, we can only take independent samples from the model until all possible placement modes are covered. In contrast, for AnyPlace and AnyPlace-EBM, the high-level VLM module predicts possible placement locations, allowing us to perform sampling for each mode separately. We perform an equal number of samples for each mode and compare performance based on the same total number of samples across all models.



(A) Vial Insertion Task



(B) Hanging Task

Figure 4: Coverage comparison across different models in vial insertion and hanging tasks. In both cases, AnyPlace achieves near-perfect coverage with just a few samples, while baselines fail to match this even with 100 samples.

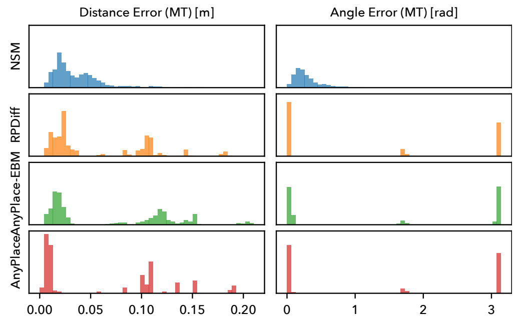


Figure 5: Errors on insertion tasks. Based on histograms of the translation and rotation error in pose prediction, it is clear that AnyPlace achieves the best precision. (MT): models trained on multi-task data.

Methods	Insert vial (10 modes)	Hang ring (5 modes)	Stack battery (3 modes)
NSM	0%	0%	0%
RPDiff	0%	60%	0%
AnyPlace-EBM	50%	0%	0%
AnyPlace	**80%**	**80%**	**67%**

Table 2: Coverage rate of the real robot executing three placement tasks. 10 trials were conducted for each task. Our model significantly outperforms baseline models in real experiments, demonstrating its ability to generalize to unseen objects and effectively handle noisy data.

In Figure 4A, we show results for the vial insertion task. AnyPlace rapidly approaches peak performance with a single sample per mode due to its high placement success rate and the VLM's strong reasoning in identifying diverse placement locations. AnyPlace-EBM models show a similar trend but plateau at 73% due to lower success rates. In contrast, RPDiff fails to capture multimodality, with coverage below 10%, and is outperformed even by the NSM regression model. For hanging, we assess generalization across racks with varied sizes, geometries, and stick spacing. The placement coverage is reported in Figure 4B. As with vial insertion, AnyPlace consistently achieves 100% coverage with fewer samples. While RPDiff improves here, its coverage saturates at 90%, revealing limited generalization. In contrast, AnyPlace 's coarse predictions guide the fine model to focus on local placement, enabling robust generalization across object variations.

Placement Precision: Fine-grained Insertion. Our final simulation evaluation focuses on assessing precision, a key factor in many placement tasks. We test whether restricting input regions improves placement pose precision by directly analyzing model predictions. Figure 5 shows distance and rotation error distributions for insertion tasks. We base the error on predicted poses to the closest viable placement pose. AnyPlace yields smaller and more consistent errors than baselines. We also evaluate rotation predictions, excluding yaw due to object symmetry. As the initial pose is randomized and models cannot determine this orientation, all methods produce uniformly random yaw angle values. Most models predict flipped and correct orientations equally, except NSM, which performs worse in both position and orientation. Both AnyPlace and RPDiff perform well on orientation, while AnyPlace-EBM achieves slightly lower accuracy than RPDiff but surpasses NSM, highlighting its potential as a non-diffusion-based alternative.

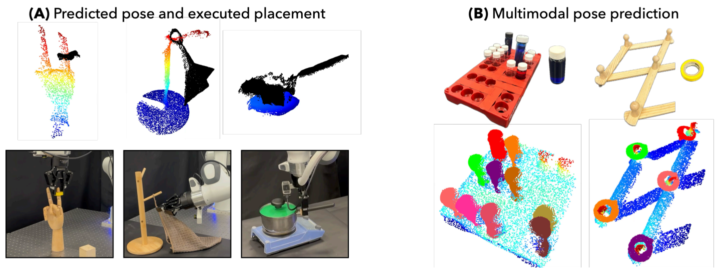

(A) Predicted pose and executed placement
(B) Multimodal pose prediction

Figure 6: Demonstration of robot executing diverse real-world tasks using AnyPlace predictions. Trained only on synthetic data, the model generalizes to unseen objects and effectively handles noisy point clouds.

5.1 Real World Evaluation

We evaluate our approach on 16 real-world placement configurations with different objects. For each scene, a single RGBD image is captured using a ZED Mini camera mounted on a Franka Emika arm. We use the same high-level pipeline and models trained solely on synthetic data to predict placement poses in a zero-shot setting. We use a simplified pipeline for executing the placement, using a specific grasp and performing placement inverse kinematics instead of a full motion planner. In addition, we do not utilize rejection sampling, but directly execute each trajectory on the real robot. Our goal is to test whether predicting local placements enables generalization to unseen objects. Additional visualization and discussion can be found in Appendix C.

Precise Multi-modal Placement. We perform systematic evaluations of coverage and success rate over 10 trials on three tasks, as shown in Figure 1, each involving multiple possible placement locations, as shown in Table 2. Overall, NSM and RPDiff fail to adapt to the differences in real-world tasks compared to the training conditions. AnyPlace, on the other hand, successfully completes placements, achieving the best performance across all tasks — 80% on the fine-grained vial insertion task. This demonstrates not only the effectiveness of the VLM in identifying placement modes, but also the generalization and precision of our low-level pose prediction model.

Generalizable & Language-conditioned Placement. To evaluate generalization, we extensively test AnyPlace on a variety of real-world placement tasks. The robot successfully performs diverse placements with unseen rigid and deformable objects, such as stacking a funnel on a holder, hanging tools and a towel on racks (Figure 1A-E). It handles fine-grained (Figure 1F-H: peg-in-hole, ring stacking) and long-horizon tasks (Figure 1N-O: lid-on-pot, followed by pot-on-stove). It can also perform placement at different locations based on language description (Figure 1K-M: bottle on different shelves). We also visualize raw pose predictions made by the model for a selection of these experiments, by moving the object point cloud to the predicted pose (Figure 6A), as well as showing multimodal predictions at the same time (Figure 6B). These experiments demonstrate that combining VLM-based location prediction with local pose refinement enables accurate and language-guided placements. This integration, along with synthetic training, allows generalization to unseen objects and complex configurations.

6 Conclusion

In this work, we presented a general pipeline for performing a wide range of object placement tasks using a robotic arm. We proposed a two-part framework consisting of a high-level module that determines coarse placement locations and a low-level module that predicts fine placement poses. The core idea of our approach is to leverage a VLM to propose placement locations, allowing the low-level pose prediction model to focus only on the local region of interest in the object's point cloud. This effectively reduces complexity and enhances generalization. To train our model, we created a synthetic dataset containing thousands of randomly generated objects and placement poses. We demonstrated the effectiveness of the entire pipeline in both simulation and real-world experiments. In simulation, we showed that AnyPlace outperforms baseline methods in terms of success rate, coverage, and precision. We then validated its robustness and generalization in real-world settings, where, given a single RGB-D image, AnyPlace predicts diverse placement configurations in a zero-shot manner and successfully generalizes to unseen objects, despite being trained purely with synthetic data.

7 Limitations

While our main focus in this work is on predicting placement poses, there are still challenges to be tackled in order to be able to execute the full pick-and-place task with the same generality. Not every stable grasp of an object can be used to realize the placement of the same object at a specific pose, and performing rejection sampling in real-world scenarios can be difficult and time-consuming. We do believe our synthetic dataset and evaluation pipeline provide a great foundation for making progress in this direction, by using them to generate data for training an end-to-end pick-and-place model applicable to a wide range of real-world placement tasks.

A common limitation of object rearrangement approaches, similar to AnyPlace, is their lack of consideration for object contact interactions during placement. By only controlling the robot end-effector placement pose based on our model prediction, the robot approaches objects too aggressively, leading to hard impacts or failed placements due to a lack of smooth and soft contact. Nevertheless, we demonstrate that AnyPlace is capable of handling a diverse set of placement tasks in the real world. A promising direction for future work is to incorporate compliance control or force feedback to adapt the robot's motion during contact and achieve more reliable placements.

While our approach enables greater precision in the placement pose prediction, it is still limited by the precision of the point clouds it receives as input. Completing placement tasks requiring significant precision with imperfect point cloud data can be very difficult. Specifically, we visualize the object point clouds at the predicted placement poses for failure cases (see Appendix Figure A4). For ring hanging, the point cloud of the rack is too sparse, causing the predicted position to be off, although the orientation remains mostly correct. For peg inserting, the captured point clouds of the holes are noisy and have vague boundaries. While our model predicts the correct orientation, it slightly misses the hole, leading to insertion failure. For battery stacking, the point cloud of the battery is largely incomplete, causing the predicted placement to be tilted rather than vertical. The battery falls over after the robot releases it. For future work, recent methods for estimating depth from RGB images have some potential to aid in this, as does continued progress in sensor quality. Another very promising approach for tackling this problem is having a policy for performing the final section of the placement based on force/torque feedback. Similarly, like in the previous point, we also think that our dataset and simulation pipeline can be a great starting point for tackling this issue, by providing us with data for training such reactive placement-execution policies.

The approach we are proposing also does not include language conditioning in the low-level pose prediction model. This means that we are unable to choose between different types of placements in the same location. However, with the architecture of the low-level model, it is straightforward to add an additional language-conditioning input. Our synthetic data generation pipeline lends itself well to data generation in this case as well, where we can automatically add language conditioning to different placement types and even use an LLM to add diversity to language-conditioning data.

Acknowledgments

This research was undertaken thanks in part to funding provided to the University of Toronto's Acceleration Consortium from the Canada First Research Excellence Fund, grant number CFREF-2022-00042. We acknowledge the generous support of Dr. Anders G. Frøseth, the Acceleration Consortium, the Vector Institute, Natural Resources Canada and the Canada 150 Research Chairs program.

References

[1] A. Simeonov, A. Goyal, L. Manuelli, L. Yen-Chen, A. Sarmiento, A. Rodriguez, P. Agrawal, and D. Fox. Shelving, stacking, hanging: Relational pose diffusion for multi-modal rearrangement. *Conference on Robot Learning*, 2023.

[2] 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, 2021. URL https://arxiv.org/abs/2112.05124.

[3] E. Chun, Y. Du, A. Simeonov, T. Lozano-Perez, and L. Kaelbling. Local neural descriptor fields: Locally conditioned object representations for manipulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1830–1836, 2023. doi:10.1109/ICRA48891.2023.10160423.

[4] A. Simeonov, Y. Du, Y.-C. Lin, A. R. Garcia, L. P. Kaelbling, T. Lozano-Pérez, and P. Agrawal. Se(3)-equivariant relational rearrangement with neural descriptor fields. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 835–846. PMLR, 14–18 Dec 2023. URL `https://proceedings.mlr.press/v205/simeonov23a.html`.

[5] C. Pan, B. Okorn, H. Zhang, B. Eisner, and D. Held. TAX-pose: Task-specific cross-pose estimation for robot manipulation. In *6th Annual Conference on Robot Learning*, 2022. URL `https://arxiv.org/abs/2211.09325`.

[6] H. Ryu, J. Kim, H. An, J. Chang, J. Seo, T. Kim, Y. Kim, C. Hwang, J. Choi, and R. Horowitz. Diffusion-edfs: Bi-equivariant denoising generative modeling on se(3) for visual robotic manipulation, 2023. URL `https://arxiv.org/abs/2309.02685`.

[7] H. Ryu, H. in Lee, J.-H. Lee, and J. Choi. Equivariant descriptor fields: Se(3)-equivariant energy-based models for end-to-end visual robotic manipulation learning, 2023. URL `https://arxiv.org/abs/2206.08321`.

[8] C. Gao, Z. Xue, S. Deng, T. Liang, S. Yang, L. Shao, and H. Xu. Riemann: Near real-time se(3)-equivariant robot manipulation without point cloud segmentation. 2024.

[9] B. Eisner, Y. Yang, T. Davchev, M. Vecerik, J. Scholz, and D. Held. Deep SE(3)-equivariant geometric reasoning for precise placement tasks. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://arxiv.org/abs/2404.13478`.

[10] H. Huang, K. Schmeckpeper, D. Wang, O. Biza, Y. Qian, H. Liu, M. Jia, R. Platt, and R. Walters. Imagination policy: Using generative point cloud models for learning manipulation policies. *arXiv preprint arXiv:2406.11740*, 2024.

[11] H. Huang, H. Liu, D. Wang, R. Walters, and R. Platt. Match policy: A simple pipeline from point cloud registration to manipulation policies, 2024. URL `https://arxiv.org/abs/2409.15517`.

[12] Y.-C. Chen, H. Li, D. Turpin, A. Jacobson, and A. Garg. Neural shape mating: Self-supervised object assembly with adversarial shape priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12724–12733, 2022.

[13] Y. Ding, H. Geng, C. Xu, X. Fang, J. Zhang, S. Wei, Q. Dai, Z. Zhang, and H. Wang. Open6DOR: Benchmarking open-instruction 6-dof object rearrangement and a VLM-based approach. In *First Vision and Language for Autonomous Driving and Robotics Workshop*, 2024. URL `https://openreview.net/forum?id=RclUiexKMt`.

[14] W. Yuan, A. Murali, A. Mousavian, and D. Fox. M2t2: Multi-task masked transformer for object-centric pick and place. In *7th Annual Conference on Robot Learning*, 2023.

[15] Z. He, N. Chavan-Dafle, J. Huh, S. Song, and V. Isler. Pick2place: Task-aware 6dof grasp estimation via object-centric perspective affordance. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7996–8002, 2023. doi:10.1109/ICRA48891.2023.10160736.

[16] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox. Rvt: Robotic view transformer for 3d object manipulation. *arXiv:2306.14896*, 2023.

[17] A. Goyal, V. Blukis, J. Xu, Y. Guo, Y.-W. Chao, and D. Fox. Rvt2: Learning precise manipulation from few demonstrations. *RSS*, 2024.

[18] S. James, K. Wada, T. Laidlow, and A. J. Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation, 2022. URL https://arxiv.org/abs/2106.12534.

[19] B. Chen, Z. Xu, S. Kirmani, B. Ichter, D. Sadigh, L. Guibas, and F. Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14455–14465, June 2024.

[20] M. Deitke, C. Clark, S. Lee, R. Tripathi, Y. Yang, J. S. Park, M. Salehi, N. Muennighoff, K. Lo, L. Soldaini, J. Lu, T. Anderson, E. Bransom, K. Ehsani, H. Ngo, Y. Chen, A. Patel, M. Yatskar, C. Callison-Burch, A. Head, R. Hendrix, F. Bastani, E. VanderBilt, N. Lambert, Y. Chou, A. Chheda, J. Sparks, S. Skjonsberg, M. Schmitz, A. Sarnat, B. Bischoff, P. Walsh, C. Newell, P. Wolters, T. Gupta, K.-H. Zeng, J. Borchardt, D. Groeneveld, J. Dumas, C. Nam, S. Lebrecht, C. Wittlif, C. Schoenick, O. Michel, R. Krishna, L. Weihs, N. A. Smith, H. Hajishirzi, R. Girshick, A. Farhadi, and A. Kembhavi. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models, 2024. URL https://arxiv.org/abs/2409.17146.

[21] G. R. Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl, S. Bohez, K. Bousmalis, A. Brohan, T. Buschmann, A. Byravan, S. Cabi, K. Caluwaerts, F. Casarini, O. Chang, J. E. Chen, X. Chen, H.-T. L. Chiang, K. Choromanski, D. D'Ambrosio, S. Dasari, T. Davchev, C. Devin, N. D. Palo, T. Ding, A. Dostmohamed, D. Driess, Y. Du, D. Dwibedi, M. Elabd, C. Fantacci, C. Fong, E. Frey, C. Fu, M. Giustina, K. Gopalakrishnan, L. Graesser, L. Hasenclever, N. Heess, B. Hernaez, A. Herzog, R. A. Hofer, J. Humplik, A. Iscen, M. G. Jacob, D. Jain, R. Julian, D. Kalashnikov, M. E. Karagozler, S. Karp, C. Kew, J. Kirkland, S. Kirmani, Y. Kuang, T. Lampe, A. Laurens, I. Leal, A. X. Lee, T.-W. E. Lee, J. Liang, Y. Lin, S. Maddineni, A. Majumdar, A. H. Michaely, R. Moreno, M. Neunert, F. Nori, C. Parada, E. Parisotto, P. Pastor, A. Pooley, K. Rao, K. Reymann, D. Sadigh, S. Saliceti, P. Sanketi, P. Sermanet, D. Shah, M. Sharma, K. Shea, C. Shu, V. Sindhwani, S. Singh, R. Soricut, J. T. Springenberg, R. Sterneck, R. Surdulescu, J. Tan, J. Tompson, V. Vanhoucke, J. Varley, G. Vesom, G. Vezzani, O. Vinyals, A. Wahid, S. Welker, P. Wohlhart, F. Xia, T. Xiao, A. Xie, J. Xie, P. Xu, S. Xu, Y. Xu, Z. Xu, Y. Yang, R. Yao, S. Yaroshenko, W. Yu, W. Yuan, J. Zhang, T. Zhang, A. Zhou, and Y. Zhou. Gemini robotics: Bringing ai into the physical world, 2025. URL https://arxiv.org/abs/2503.20020.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[23] H.-S. Fang, C. Wang, H. Fang, M. Gou, J. Liu, H. Yan, W. Liu, Y. Xie, and C. Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *IEEE Transactions on Robotics (T-RO)*, 2023.

[24] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. V. Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, N. Ratliff, and D. Fox. curobo: Parallelized collision-free minimum-jerk robot motion generation, 2023.

[25] Y. You, L. Shao, T. Migimatsu, and J. Bohg. Omnihang: Learning to hang arbitrary objects using contact point correspondences and neural collision estimation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[26] K. A. Murphy, C. Esteves, V. Jampani, S. Ramalingam, and A. Makadia. Implicit-pdf: Non-parametric representation of probability distributions on the rotation manifold. In *Proceedings of the 38th International Conference on Machine Learning*, pages 7882–7893, 2021.

[27] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi:10.1109/LRA.2023.3270034.

[28] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[29] I. Loshchilov and F. Hutter. Decoupled weight decay regularization, 2019. URL `https://arxiv.org/abs/1711.05101`.

A Language Prompts for Molmo and Predicted Placement Visualization

To extract object point clouds, we first prompt Molmo (e.g., points to the bottle) to predict a single anchor point. This anchor is then passed to SAM-2 to obtain segmentation masks of the object. The complete point clouds are subsequently extracted using the segmentation and camera information. To predict placement locations, in Figure A1 and Figure A2, we show language prompts used by the VLM in both real-world and simulation experiments. Based on our observations, the Molmo VLM accurately identifies the correct placement locations in both real-world and simulated images based on the language input. Even when the predicted location is not perfectly centered, our low-level pose prediction model can still be successful in predicting the placement pose. For instance, when predicting the placement of the bottle on the top shelf (Figure A1, last image in the top row), the VLM may give a location at the very edge. However, our pose-prediction model focuses on the entire local region extracted around that point and is able to provide a pose where the entire object is on the surface, allowing the robot to execute the task successfully.

We do notice that minor prompt engineering is occasionally needed to generate accurate predictions using Molmo, such as “point to the empty slot between the two poles”. We believe this limitation can be addressed as VLM models continue to improve.

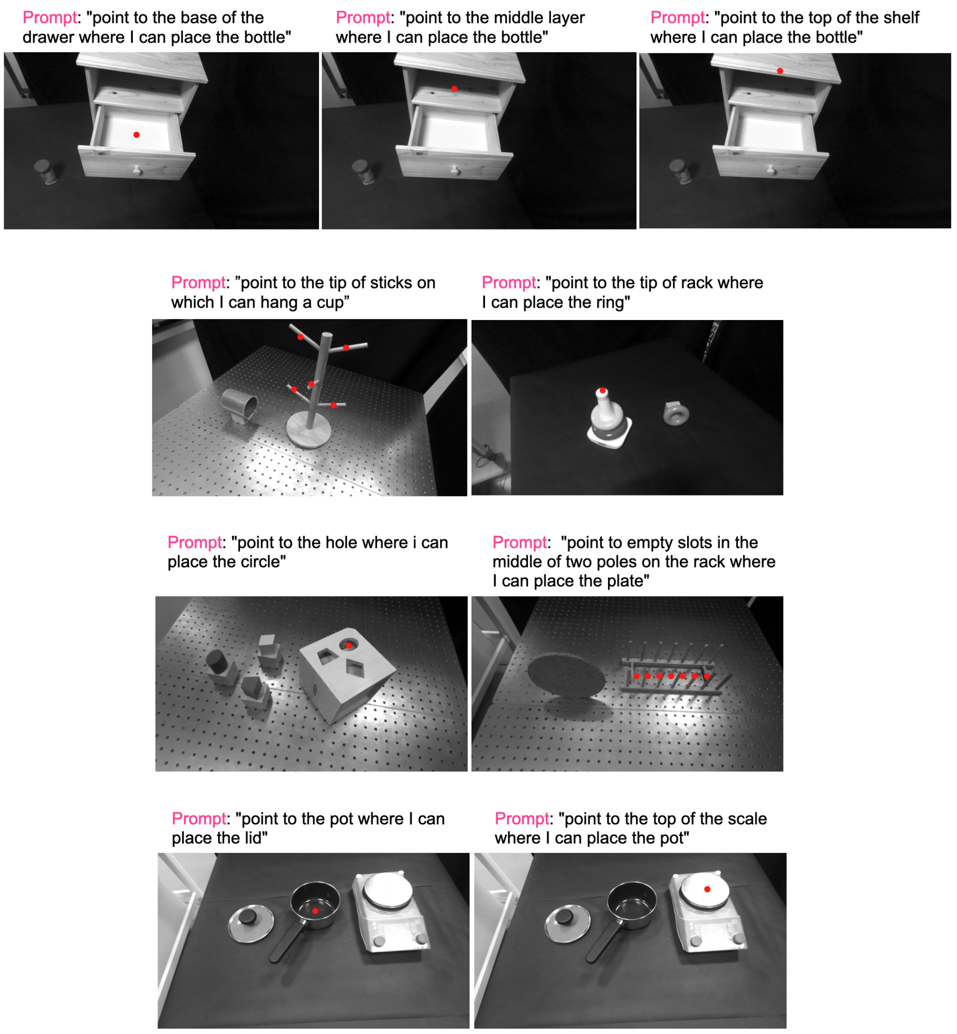

Prompt: "point to the base of the drawer where I can place the bottle"
Prompt: "point to the middle layer where I can place the bottle"
Prompt: "point to the top of the shelf where I can place the bottle"
Prompt: "point to the tip of sticks on which I can hang a cup”
Prompt: "point to the tip of rack where I can place the ring"
Prompt: "point to the hole where i can place the circle"
Prompt: "point to empty slots in the middle of two poles on the rack where I can place the plate"
Prompt: "point to the pot where I can place the lid"
Prompt: "point to the top of the scale where I can place the pot"

Figure A1: Additional Language Prompts and VLM Output Visualization in the Real-World Evaluation.

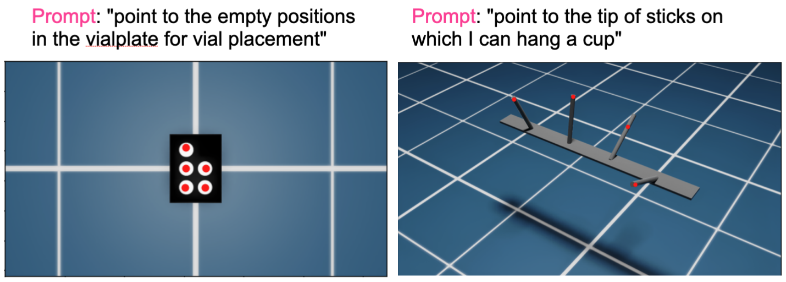

Prompt: "point to the empty positions in the vialplate for vial placement"
Prompt: "point to the tip of sticks on which I can hang a cup"

Figure A2: Language Prompts and VLM Output Visualization in the Simulation Evaluation. In our simulation experiments, we use the VLM to generate placement locations on RGB images from the simulator. Based on this proposed location, we crop the point clouds and input them into our placement pose prediction model.

B Additional Details on the Low-level Diffusion Model

B.1 Model Architecture

For our diffusion-based placement prediction model, we leverage a transformer architecture. Details of the model architecture are listed in Table 3. Starting with two object point clouds, the diffusion process iteratively denoises the relative transformation, gradually moving the object being placed toward its final pose. Initially, we transform the object point cloud $\mathcal{P}_{\text{c}}$ with a random transformation $T_{\text{init}} = (\mathbf{R}, \mathbf{t})$ to get $\mathcal{P}_{\text{c}}^{(0)}$, where R is randomly sampled over the SO(3) space, and t is sampled within the bounding box of the cropped placement region:

$$\mathcal{P}_c^{(0)} = T_{\text{init}}\, \mathcal{P}_c, \tag{1}$$

where

$$T_{\text{init}} = (\mathbf{R}, \mathbf{t}), \quad \mathbf{R} \sim \mathcal{U}(\text{SO}(3)), \quad \mathbf{t} \sim \mathcal{U}(\text{bbox}(\mathcal{P}_{b\_crop})). \tag{2}$$

As shown in Figure 2, at each denoising timestep t, $\mathcal{P}_{\text{c}}^{(\text{t})}$ and $\mathcal{P}_{\text{b\_crop}}$ are input into the encoder. Specifically, both point clouds are first downsampled to 1024 points using Farthest Point Sampling (FPS) and normalized to the size of a unit cube. The downsampled point clouds are passed through a linear layer to extract latent features, which are subsequently concatenated with a one-hot vector used to identify the corresponding point cloud. These combined features are then processed by the Transformer encoder [22, 12], where self-attention layers are applied to effectively extract features from the point clouds. We then leverage cross-attention and pooling layers to further aggregate these features, producing a unified feature representation that captures the spatial relationship between the two objects. In the decoder, we first obtain the sinusoidal positional embedding of the diffusion timestep t. Finally, the joint point cloud feature representation, along with the encoded timestep, is fed into MLP layers to predict the relative transformation $T_n^{(t)}$ consisting of a rotation $\mathbf{R} \in \text{SO}(3)$ and a translation $\mathbf{t} \in \mathbb{R}^3$ for refining the object's pose. The target object point clouds are then transformed accordingly before proceeding to the next denoising step as $\mathcal{P}_{\text{c}}^{(\text{t-1})} = T_n^{(t)} \mathcal{P}_{\text{c}}^{(\text{t})}$. The full transformation T_n taking the object from its initial location to the placement pose is the product of all the incremental transformations predicted through the diffusion steps: $T_n = \prod_{t=1}^{t_{\max}} T_n^{(t)}$.

We do not utilize a learned classifier on top of the pose prediction model that is present in RPDiff. Such a model can be applied on top of any of the methods we evaluate here, including the ones we propose. Not having it also allows us to evaluate the number of samples needed to achieve a particular coverage of possible placement locations in the scene.

B.2 Model Training

During training of our low-level diffusion pose prediction model, we perform 5 denoising steps. Instead of incrementally adding Gaussian noise to the input during the forward process, as is common practice [28], we manually define the noise added at each timestep. In our case, the noise is the relative transformation that the model predicts. Specifically, the intermediate ground truth relative

Table 3: Summary of Model Architecture

Model Component	**Details**
Model Total Parameters	4,279,688
Number of Heads	1
Number of Self-attention Blocks	4
Number of Cross-attention Blocks	4
Point Cloud Feature Dimension	258 (256 + one-hot embedding)
Transformer Feature Dimension	256
Encoder	
Self-Attention	Multi-Headed Attention (1 head)
Feedforward Layer	Linear (258 $\rightarrow$ 256), ReLU, Linear (256 $\rightarrow$ 258)
Normalization	LayerNorm
Decoder	
Self-Attention	Multi-Headed Attention (1 head)
Cross-Attention	Multi-Headed Attention (1 head)
Feedforward Layer	Linear (258 $\rightarrow$ 256), ReLU, Linear (256 $\rightarrow$ 258)
Normalization	LayerNorm

transformations $T^{(t)}_{n,GT}$ are generated by linearly interpolating the translation and using spherical linear interpolation (SLERP) to sample rotations between the object's initial and final placement poses. During inference, we generate diverse placement poses by sampling the diffusion model multiple times, each time starting with randomly transformed initial object point clouds $\mathcal{P}_{\mathrm{c}}^{(0)}$. We perform a larger number of denoising steps at test time, by repeating the last denoising step more times for a total of 50 denoising steps, similar to RPDiff [1]. All single-task models are trained for three days, while multitask models are trained for five days on a single NVIDIA A100 GPU. Other parameters can be found in Table 4.

Table 4: Parameters for Model Training

Parameter	**Value**
Diffusion steps	5
Number of training iteration	500k
Batch size	48
Optimizer	AdamW [29]
Learning rate	1×10^{-4}
Learning rate schedule	Linear warmup and cosine decay
Warmup epochs	50
Weight decay	0.1
Optimizer momentum	$\beta_1 = 0.9$, $\beta_2 = 0.95$

B.3 Additional Details on Data Generation

For object generation in Blender, we procedurally generate 3D objects, such as pegs, holes, cups, racks, beakers, vials, and vial holders. Object parameters—including height, width, length, shapes, and number of edges—are randomized to increase variability. For racks and vial plates, we also randomize the number of poles and holes. Additionally, random scaling is applied along the x, y, and z axes. This type of programmatic object generation essentially gives us placement poses ”for free”.

This approach for object generation not only allows simple randomization of object geometries, but also supports a range of valid placements rather than a single fixed pose. For training the AnyPlace low-level pose-prediction model, only local placement data is needed, so we do not have to generate full objects with all possible placement modes, further simplifying scalability. While adding completely new object classes does require some human effort, it is limited to writing a single piece of code per class, and common geometric aspects can often be shared. Overall, building a large dataset of parametrized object classes enables programmatic creation of vast amounts of synthetic training data, supporting not only placement tasks but potentially many other manipulation problems.

To identify stable placements for objects, we use NVIDIA IsaacSim to determine object placement poses. At the start of each trial, two objects are randomly sampled and loaded into the simulation. Since all objects are procedurally generated and possible placement locations are known during generation (e.g., the center of each hole on a vial plate), the ideal object placement location for the placing object can easily be determined. This approach finds object placement locations that maximize the clearance between the objects in their final placement configurations. For object placement rotations, they are then randomly sampled along their axis of symmetry to explore various placement poses. Four cameras are set up to capture dense object point clouds and render RGBD images. This dataset covers a wide range of placement scenarios encountered in real life. Table 5 presents statistics on the number of placements generated using our synthetic dataset generation pipeline. Notably, by focusing only on the region of interest for placement pose prediction, AnyPlace models perform well across different placement tasks and generalize to unseen objects. They are trained with fewer than 2000 samples per task, demonstrating the effectiveness of our design.

Table 5: Dataset Size for Each Placement Task

Placement Task	**Number of Placements**
Hanging	1,767
Stacking	1,696
Vial Insertion	1,107
Other Insertion	800

C Evaluation

C.1 Implementation of the Pick and Place Pipeline

After determining the placement poses by AnyPlace, we implement a pick-and-place pipeline to manipulate the object and position it accurately at the target pose. Specifically, grasp detection begins by extracting the target object point clouds $\mathcal{P}_c$ from an RGBD image. AnyGrasp [23] then processes the resulting point clouds to identify the optimal grasp candidates sorted by confidence. To pick up the object, the gripper is first moved to a pre-grasp pose, positioned 10 centimeters away from the target along the gripper's z-axis. The gripper then approaches the target in a straight line while maintaining its orientation. Similarly, during placement, the robot first moves to a pre-place pose, followed by a final approach without altering the gripper orientation. The distance from the pre-place pose to the final placement pose is adjusted according to the object's size to avoid collisions during the transition to the pre-place position. With the waypoints and end-effector orientation constraints defined, we use cuRobo to generate the complete motion plan for robot pick and place while accounting for the object collision model. For each evaluation scene, we perform rejection sampling by generating 100 grasps using AnyGrasp and forming (grasp, placement pose) pairs to identify valid grasps for the specific placement pose predicted by our model that can be executed by the robot.

The pipeline itself can be used as a separate module. It makes no assumptions about the placement prediction method and is not specific in any way to other modules in our approach. This enables it to be used as a general system for evaluating placement pose prediction models. In Figure 3 we show the system being used to perform insertion, hanging, and placement tasks. Combined with the synthetic dataset we create, this gives us a complete system for comparing different models and getting systematic results of success rate, mode coverage, and precision. Both motion planning and the pick-and-place simulation itself are GPU parallelizable, making the evaluation of new models even easier.

C.2 AnyPlace Energy-based Model

Inspired by Implicit-PDF [26], this model uses the same encoder as ours, but for the decoder, instead of explicitly predicting the placement pose, it includes two separate branches: one for placement location prediction and another for predicting the placement rotation energy. During training, we encourage all placement rotations in SO(3) that result in stable placements to have low energy by predicting the unnormalized log probability of the joint distribution between the latent state from the encoder and the SO3 rotation. This approach requires only minimal changes to the model. We use a negative log-likelihood loss for the rotation as shown below:

$$\mathcal{L}_{energy}(h, R_0) = -\log \frac{\exp(f(h, R_0))}{\sum_i^N \exp(f(h, R_i))} \tag{3}$$

During training, we estimate the normalization factor by sampling. In particular, we ran the model on 4096 randomly sampled SO3 rotations. The total loss is the sum of this loss and the same L1 loss for translation:

$$\mathcal{L}_{total} = \mathcal{L}_{translation} + \mathcal{L}_{energy} \tag{4}$$

At test time, we can get the estimate of the full distribution over the rotations conditioned on the inputs to the model in the same way by randomly sampling thousands of rotations. The rotation with the lowest energy is then selected as the final placement rotation.

We found that the translation loss consistently converges faster than the energy-based one during training as a result of its smaller scale. To address this imbalance, we experimented with various ways of weighting the two loss terms when computing the total loss, which led to slight performance improvements. In contrast, for the diffusion model, losses are computed at each diffusion step, and the scales of translation and rotation errors remain more balanced, resulting in more stable training. Overall, the energy-based model offers faster inference but lower accuracy compared to the diffusion model.

D Additional Result of Placement Precision

In this section, we provide additional results on the placement precision of fine-grained insertion tasks for all models trained on single-task data. In Figure A3, we show the distribution of distance errors for each approach on the insertion tasks. For both single-task and multi-task training, we observe that AnyPlace achieves smaller errors and does so more reliably than the baselines, in terms of translation and rotation prediction.

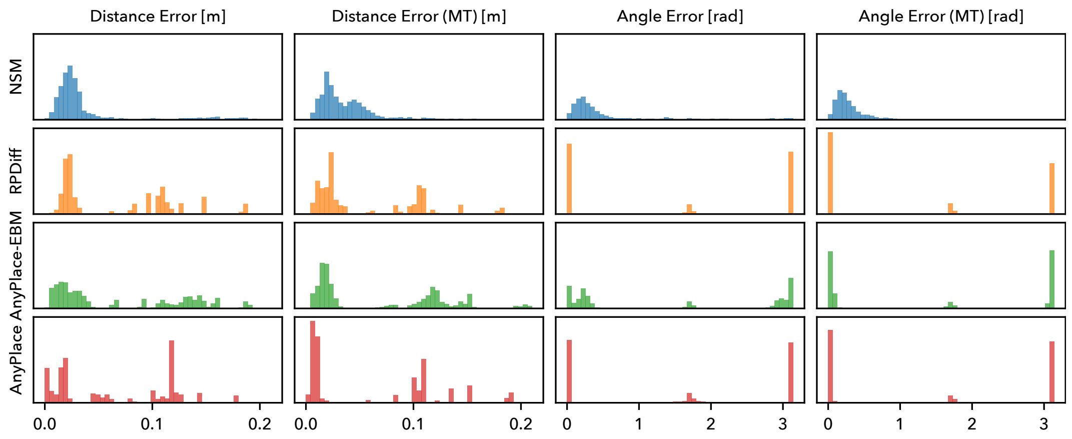


Figure A3: Evaluation of Distance and Angle Errors in Insertion Tasks. We plot histograms of the translation and rotation error in pose prediction for each approach trained on single-task data and multi-task data. We base the error on the position and rotation to the closest viable placement pose. The rotation error is in the range $[0, \pi]$. For both translation and rotation, we use 50 equally-sized bins covering the range of the variable to produce the histograms.

D.1 Additional Analysis of Real Robot Experiments

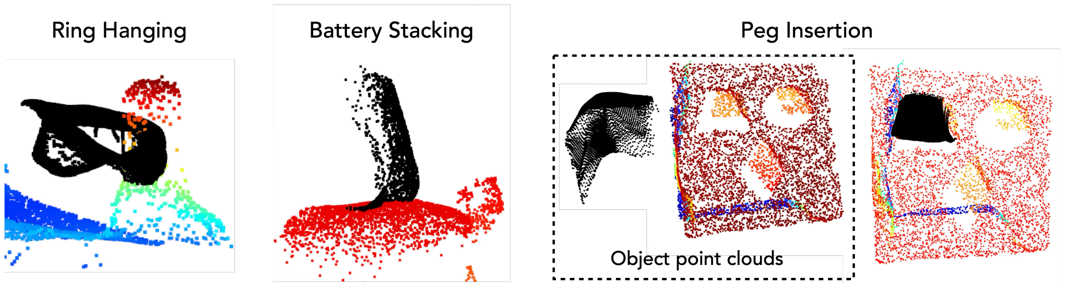


Figure A4: Object point clouds at predicted placement poses in failure cases. The point cloud of the object being placed is shown in black.

We systematically evaluate the performance of all baseline models and AnyPlace across three tasks on a real robot: vial insertion, ring hanging, and battery stacking. As shown in the Figure A5, AnyPlace successfully predicts correct and precise placement poses using partial point clouds for all tasks, due to its ability to focus on the local placement region and the robustness of its diffusion-based pose prediction model. Although AnyPlace-Energy uses the same high-level placement location module, we observe that the energy-based model often predicts incorrect poses in all tasks, both in position and orientation. Similarly, RPDiff frequently converges to the same placement pose, failing to capture diverse placement locations. It also struggles significantly with predicting accurate positions. Due to the fixed cropping size introduced in RPDiff, its generalization to various objects is limited, causing inaccurate prediction. For example, in the vial insertion and ring hanging tasks, the predicted poses often cause the placing object's point cloud to deeply intersect with the base objects. For battery stacking, the model predicts a pose that is far from the correct placement location. Finally, NSM, which uses a simple regression decoder, is unable to predict any meaningful placement poses.

To further evaluate the generalization and robustness of AnyPlace in the real world, we conducted extensive experiments across different placement scenarios. We visualize the point clouds at the predicted placement poses in Figure A6. It is clear that AnyPlace is capable of handling unseen objects and noisy point clouds. For example, in many insertion tasks, the model accurately predicts placement poses even when dealing with complex, unseen objects, such as focusing on individual fingers when hanging a ring onto a hand. This level of generalization is achieved by focusing on local regions rather than relying on global information, which may contain redundant or irrelevant details. Moreover, training the low-level model on a diverse synthetic dataset enables it to capture general concepts and representations across different placement scenarios. For example, the model was never trained with a funnel or the ring clamp, yet it successfully predicts the correct placement poses for these novel objects.

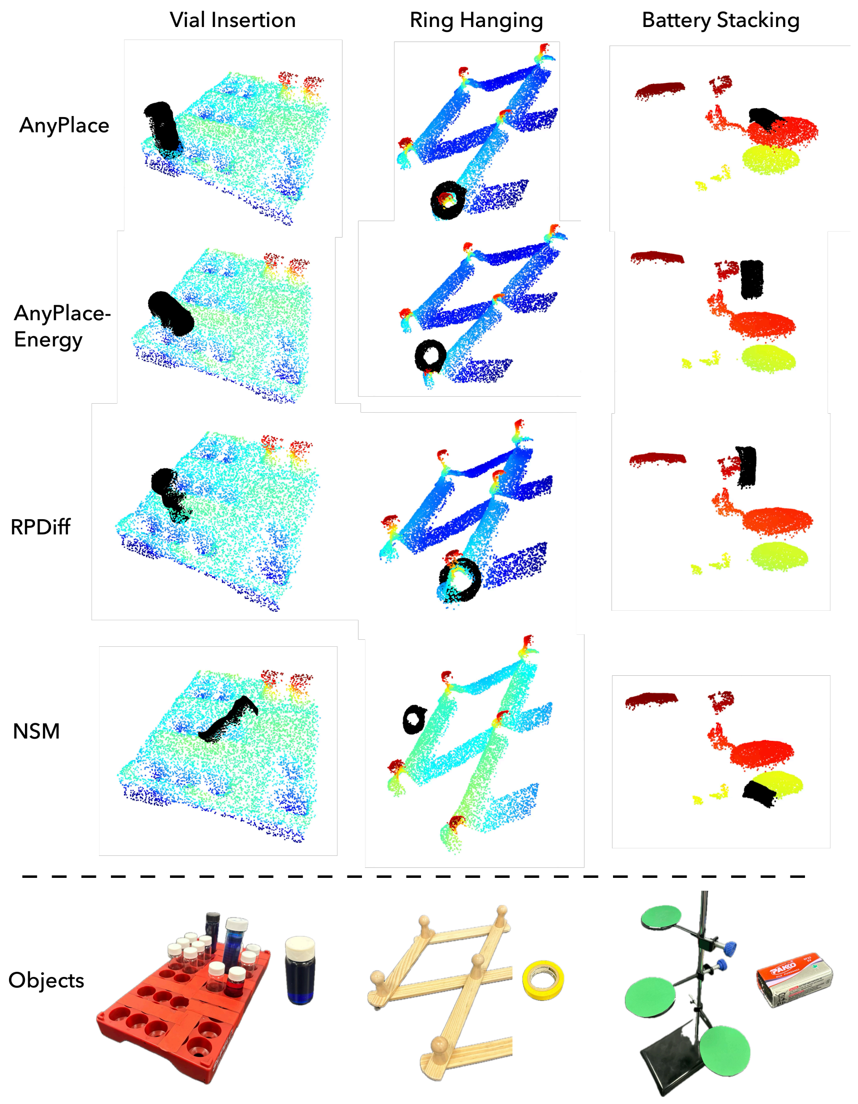


Figure A5: Visualization of Object Point Clouds at Predicted Poses from Different Models. Quantitative evaluations were conducted on three real-world tasks: vial insertion, ring hanging, and battery stacking. The figure shows examples of predicted placement poses visualized using point clouds. The point cloud of the placed object (target object) is shown in black.

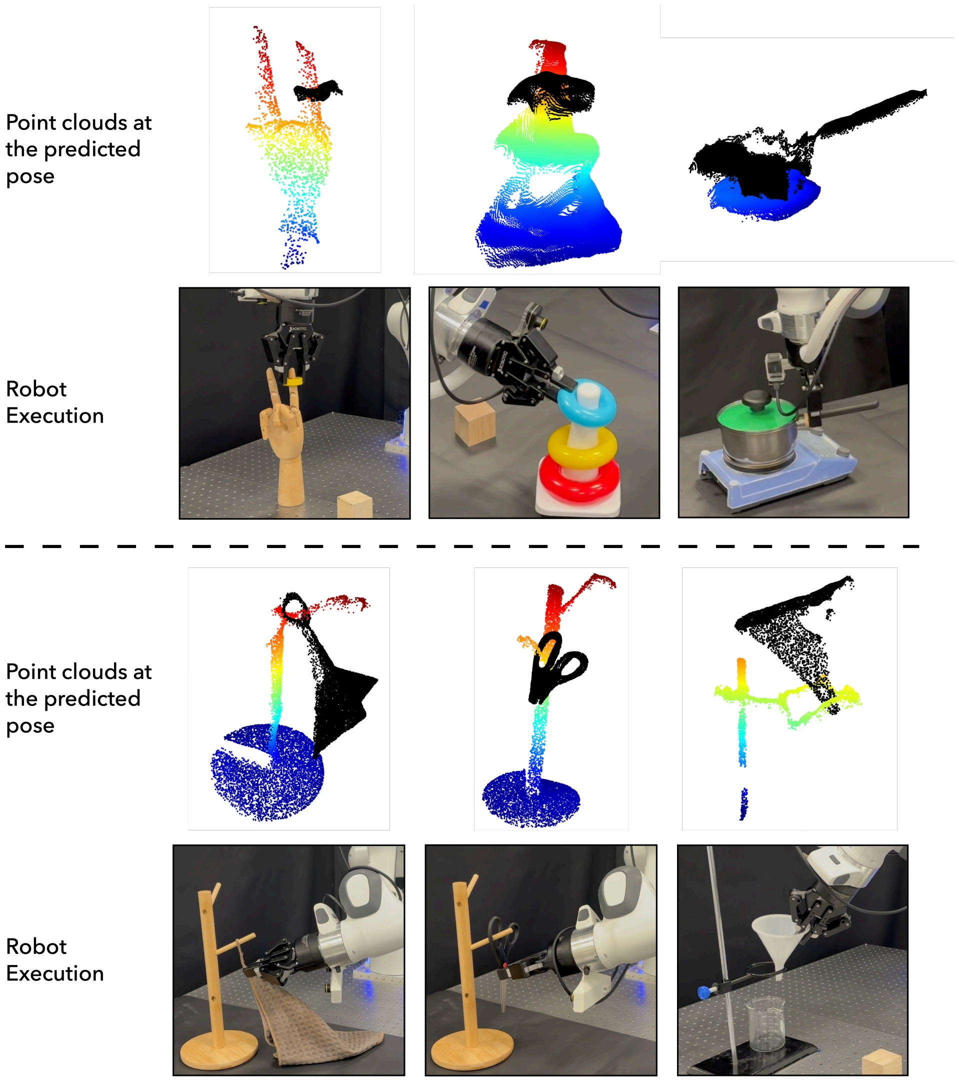

Point clouds at the predicted pose
Robot Execution
Point clouds at the predicted pose
Robot Execution

Figure A6: Object point clouds at predicted placement poses in failure cases. The point cloud of the object being placed is shown in black.