DynScene: Scalable Generation of Dynamic Robotic Manipulation Scenes for Embodied AI
– *Appendix* –

This appendix provides details for dataset preprocessing in Section A, dynamic scene generation analysis in Section B, action generation analysis in Section C, and additional ablation studies in Section D.

A. Data Preprocessing

Data Preprocessing for ARNOLD The ARNOLD dataset [8] was used as a benchmark for robotic manipulation tasks. It includes eight tasks with task-specific instructions, environments, and robot actions. Each instruction defines a goal state and related actions for the task (*e.g., open cabinet: "slide the cabinet entirely open"*). The environment data provides room layouts and object information. Robot actions are divided into four phases: the initial phase, the pre-grasp phase, the grasp phase, and the manipulation phase. Among the eight tasks, *pour water* and *transfer water* have manipulation phases consisting of three sequential actions. For instance, in the *pour water* task, the manipulation phase includes:

1. Lifting the cup to the target height for pouring.
2. Moving the cup horizontally to the pouring position.
3. Gradually tilting the cup until the goal is reached.

The remaining tasks have a single manipulation action, such as the *pickup object* task, which involves *"lifting the object to the goal height."* The data configuration follows the description provided in the main paper (Section 3.1). The static phase includes the initial environment setup, the initial phase, and task-specific instructions. The action phase encompasses the pre-grasp, grasp, and manipulation phases. To process the shape code $\boldsymbol{f}$ for each object, an autoencoder was trained, generating latent vectors that represent object point clouds, as illustrated in Figure 6.

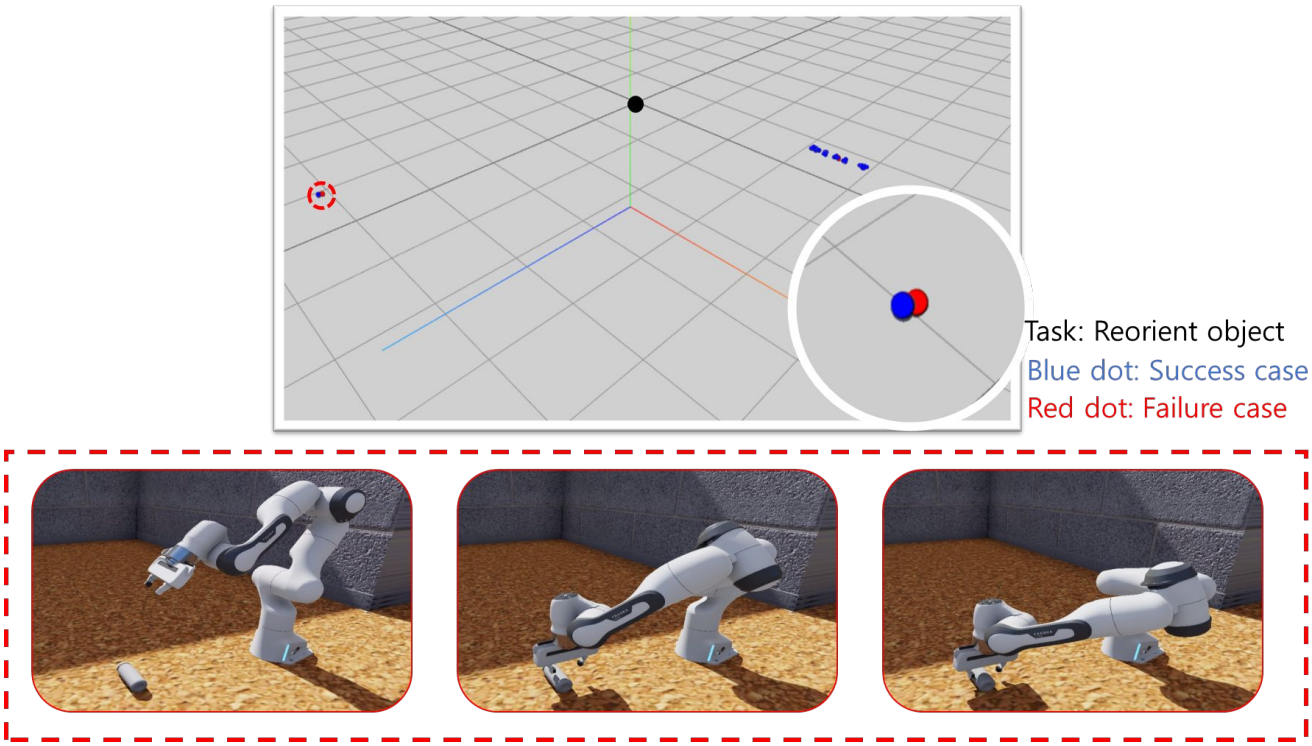
Figure 6. **Point clouds of bottle class objects**. The shape codes $\boldsymbol{f}$ are latent vectors obtained by training an autoencoder on point clouds of objects in the ARNOLD dataset. This figure illustrates how these shape codes represent the point clouds of bottle class objects.

Data Preprocessing for Generated Data For effective agent training, it is important to use only the generated dynamic scenes where commands are executed correctly. To assess data success, we used NVIDIA's Isaac Sim [21], a simulation tool that provides realistic and accurate simulations for robotics research. Figure 7 shows the Isaac Sim evaluation of 100 dynamic scenes generated by DynScen. Figures 7a and 7b illustrate the end-effector positions ($\boldsymbol{p}_2^{\text{ee}}$ and $\boldsymbol{p}_3^{\text{ee}}$) as dots during the grasp and manipulation phases of the *'reorient object'* task. To enhance clarity, all robot base positions ($\boldsymbol{p}^{\text{base}}$) are projected to the origin. In this projection, black points represent the robot's base positions, red points indicate failures identified by Isaac Sim, and blue points denote successful executions. The simulation images below each figure show the scenes corresponding to the red points. Figure 7a shows an episode where the robot fails to grasp a bottle, leading to task failure. Figure 7b shows an episode where the robot grasps the bottle but collides with a nearby shelf, causing the bottle to drop and the task to fail. Through this evaluation, only correctly executed data were used for agent training.

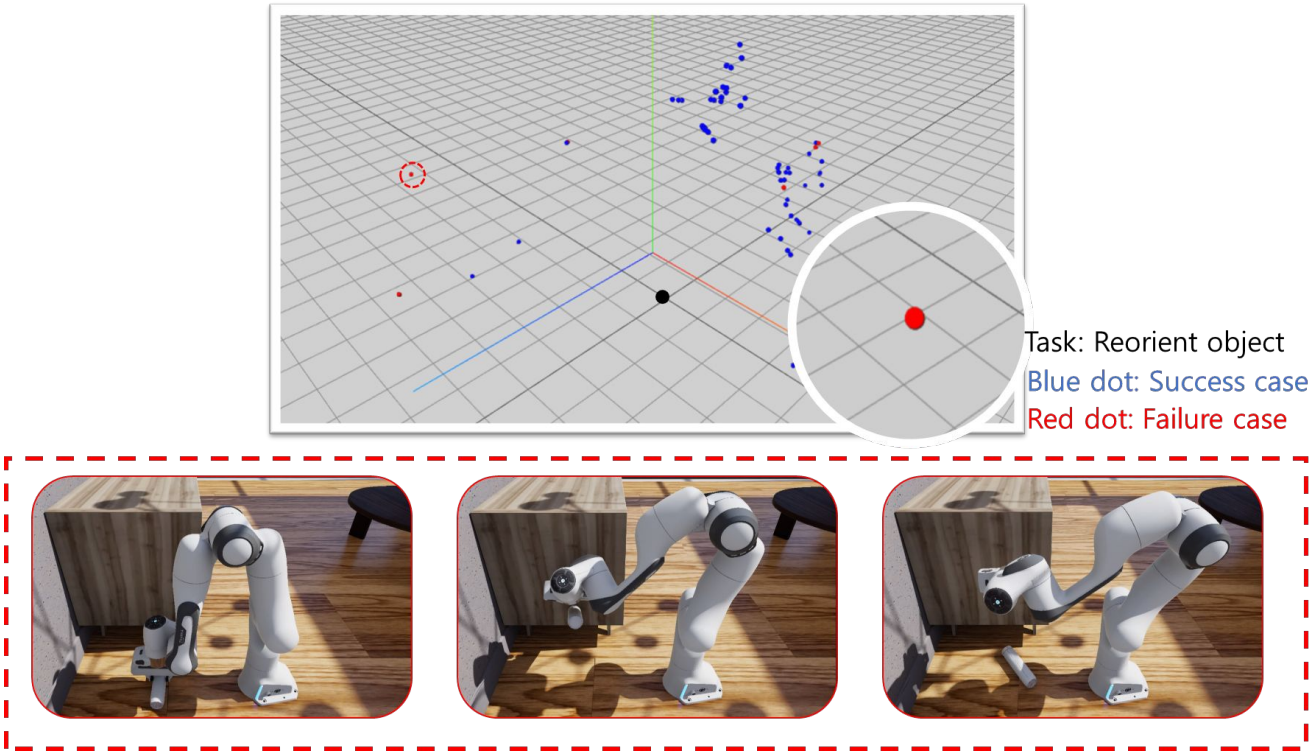
B. Dynamic Scene Generation Analysis

Diversity Evaluation for Static Scenes Section 4.2 of the main paper reports the results from 10 generated dynamic scenes for five random text prompts. Table 11 presents the text instructions (or prompts) for each task. Table 13 shows that the entropy of the generated scenes is higher than the ARNOLD dataset. The entropy measures the diversity of the static scenes, focusing on the variety of object types and the diversity of sampled layouts. This result indicates that the generated static scenes include more diverse combinations of objects and layouts than training data.

Diversity Evaluation for Actions Figure 8 visualizes the diversity of actions along the $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ axes for each task. Blue and red dots represent the mean and standard deviation of actions from DynScene and the ARNOLD dataset, respectively. DynScene exhibits greater diversity than ARNOLD across most tasks, indicating that its generated actions span a broader range of end-effector positions compared to training actions from ARNOLD. This suggests that DynScene's actions are more varied and not confined to specific trajectories. The graphs are derived from succeeded data validated by the simulator, reflecting achieved goal states. For instance, in the *pickup object* task, the y-axis graph illustrates goal states transitioning from the second to the third stage, with heights ranging from 10 to 30 cm. In contrast, tasks involving *open* actions, such as *open drawer* and *open cabinet*, show lower diversity due to a reduced success rate in generating dy-



(a) **Grasp failure in reorient Object Task**. The robot fails to grasp a bottle, resulting in the termination of the scene without task completion.



(b) **Manipulation failure in reorient object task**. Although the robot successfully grasps the bottle, it collides with a nearby shelf during manipulation, causing the bottle to be dropped and the task to fail.

Figure 7. **Failure cases in the reorient object task evaluated by Isaac Sim**. The figures illustrate failure episodes during the grasp and manipulation phases. End-effector positions ($\boldsymbol{p}_2^{\text{ee}}$ and $\boldsymbol{p}_3^{\text{ee}}$) are shown as dots, with robot base positions ($\boldsymbol{p}^{\text{base}}$) projected to the origin for clarity (black points). Red points indicate failures identified by Isaac Sim, and blue points denote successful executions. The simulation images below each subfigure correspond to the red points (failed actions).

Table 11. **Text prompts (instructions) for each task**

Task	Prompt	Task	Prompt
Pickup Object	"increase the height of the bottle ten centimeters above the ground" "pick up the bottle ten centimeters above the ground" "raise the bottle thirty centimeters from the ground" "pick up the bottle twenty centimeters from the ground" "put the bottle ten centimeters above the ground"	Reorient Object	"angle the bottle 45 degrees away from the high axis" "tilt the bottle about 180 degrees away from the upward axis" "tilt the bottle 45 degrees away from the upward axis" "the bottle is 0 degrees away from the axis of elevation" "tilt the bottle one hundred and eighty degrees away from the up axis"
Open Drawer	"open the top drawer fifty percent" "pull the middle drawer one hundred percent open" "slide the top left drawer twenty five percent open" "drag the top left dresser twenty-five percent open" "open the top left dresser 50%"	Close Drawer	"adjust the top dresser a quarter closed" "make the top left drawer seventy five percent open" "adjust the top left drawer half open" "make the top drawer 0% open" "adjust the top left drawer 50% open"
Open Cabinet	"open the cabinet a quarter" "slide the cabinet entirely open" "slide the closet midway" "open the cabinet entirely open" "slide the cabinet two quarters"	Close Cabinet	"adjust the cabinet closed completely" "shut the cupboard midway open" "adjust the closet twenty five percent open" "adjust the cabinet seventy five percent closed" "close the cupboard twenty five percent open"
Pour Water	"dump fifty percent water from the glass" "pour one hundred percent water out of the cup" "remove seventy-five percent liquid from the glass" "remove one hundred percent water out of the cup" "remove seventy-five percent water from the glass"	Transfer Water	"transfer eighty percent of the liquid to the glass" "add 80% of the liquid to the cup" "add twenty percent of the liquid to the cup" "put eighty percent of the liquid to the cup" "transfer eighty percent of the water to the cup"

Table 12. **Success rates of DynScen$_{\text{action}}$ trained on different datasets**. Comparison of success rates (%) for various tasks when DynScen$_{\text{action}}$ is trained on the ARNOLD dataset alone versus a combined dataset of DynScen-generated data and ARNOLD.

Train Dataset	Pickup Object	Reorient Object	Open Drawer	Close Drawer	Open Cabinet	Close Cabinet	Pour Water	Transfer Water	Average
ARNOLD	20.00	**80.00**	20.00	**80.00**	10.00	40.00	41.67	**30.00**	40.20
DynScen + ARNOLD	**70.00**	75.00	**35.00**	75.00	**20.00**	**45.00**	**73.33**	13.33	**50.83**

Table 13. **Entropy comparison on robotic manipulation tasks.** We compare ARNOLD and DynScen on object entropy and scene entropy across various robotic manipulation tasks.

Task	Object Entropy ↑		Scene Entropy ↑	
	ARNOLD [8]	**DynScen**	ARNOLD [8]	**DynScen**
Pickup Object	0.61	**0.64**	0.74	**0.77**
Reorient Object	0.13	**0.20**	0.21	**0.25**
Open Drawer	0.25	**0.39**	0.32	**0.50**
Close Drawer	**0.14**	**0.14**	**0.26**	**0.26**
Open Cabinet	0.23	**0.40**	0.23	**0.40**
Close Cabinet	0.66	**0.76**	0.73	**0.84**
Pour Water	0.76	**0.90**	0.57	**0.78**
Transfer Water	0.25	**0.30**	0.25	**0.30**
Average	0.38	**0.47**	0.41	**0.51**

namic scenes compared to other tasks. For all other tasks, DynScene consistently demonstrates a wider coverage of action diversity.

C. Action Generation Analysis

Visualization of Action Augmentation Dynamic scenes consist of static scenes and actions, enabling action augmentation to create new datasets by combining these components. Figure 9 displays the end-effector positions of generated actions (blue dots) and augmented actions (red dots) for the *close drawer* task, derived from the same static scene. To enhance clarity, all robot base positions ($\boldsymbol{p}^{\text{base}}$) are projected to the origin, shown as black dots. The generated actions, marked in blue, achieve a goal state of 75%, while the augmented actions, marked in red, reach a new goal state of 25%, as depicted in their respective dashed boxes. Simulation screens within these boxes highlight one successfully generated action and one augmented action. Both are selected from 100 dynamic scenes augmented from 10 original scenes and validated by Isaac Sim for success. This augmentation, leveraging residual coordinates learned from static scenes, effectively expands goal state diversity with limited data.

Action Generation as Robot Agent The actions of DynScene can function as an agent, referred to as DynScene$_{\text{action}}$, utilizing initial object and robot information to generate task-specific actions. DynScene$_{\text{action}}$ was trained in the same manner as previous agent training, using the ARNOLD dataset and a combined dataset of ARNOLD and DynScene-generated data. The evaluation was carried out using the ARNOLD test dataset [8], which was also employed for agent assessment in the main paper. Since the inputs to the PerAct [32] and DynScene$_{\text{action}}$ differ with DynScene$_{\text{action}}$ using initial object information direct numerical comparisons between the two models are not possible. Table 12 presents that training with both DynScene and ARNOLD data leads to superior outcomes compared to training solely on the ARNOLD dataset. In particular, success rates improved from 40% to 50%, demonstrating that the model generates more generalized actions across tasks when trained with dynamic scenes produced by DynScene.

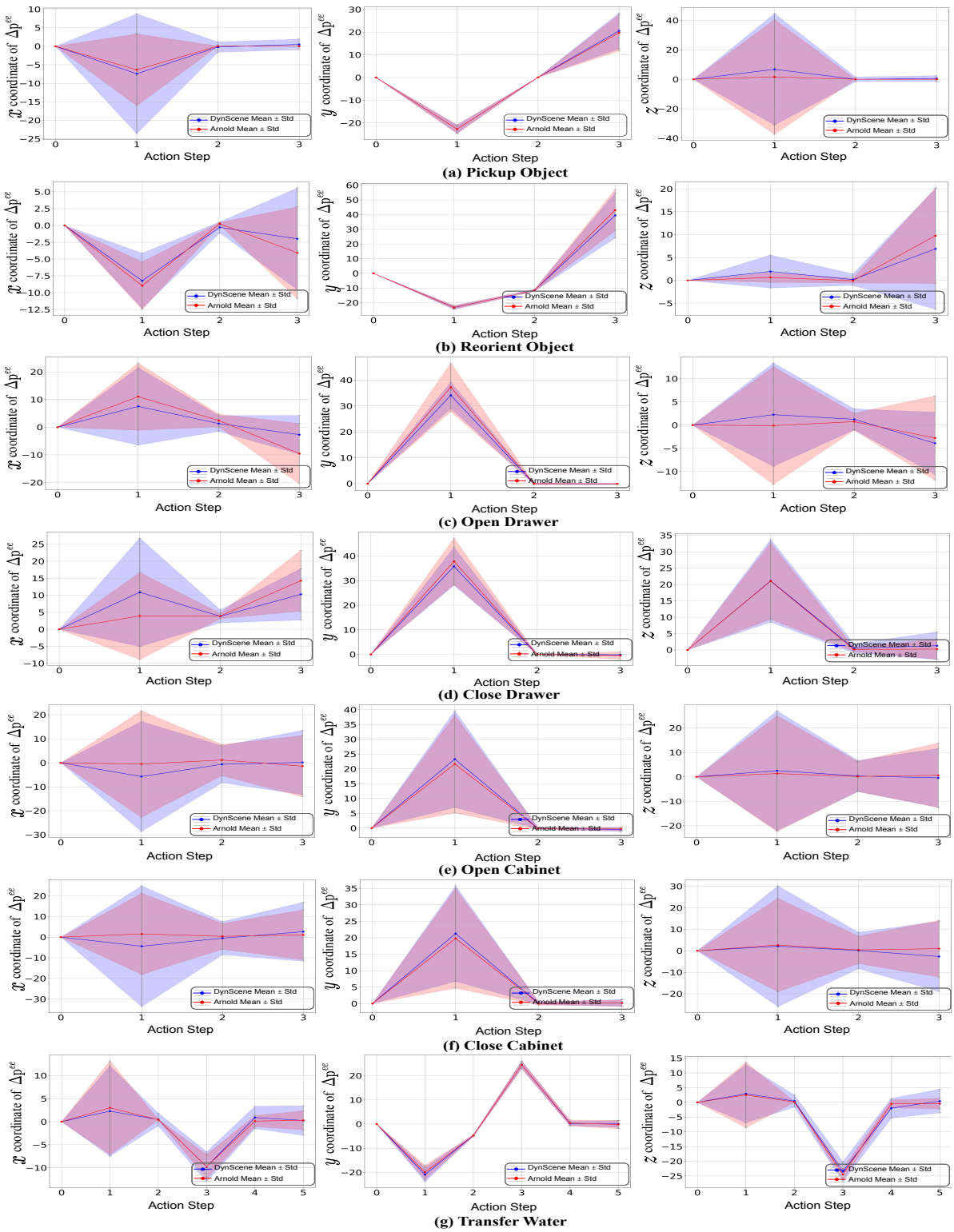


Figure 8. **Action diversity per task.** We analyze the diversity of generated actions (blue) and augmented actions (red) for all tasks except *pour water*, based on the mean and standard deviation of delta changes along the x, y, z axes. Tasks excluding *open drawer* and *open cabinet* show a wider range of action diversity.

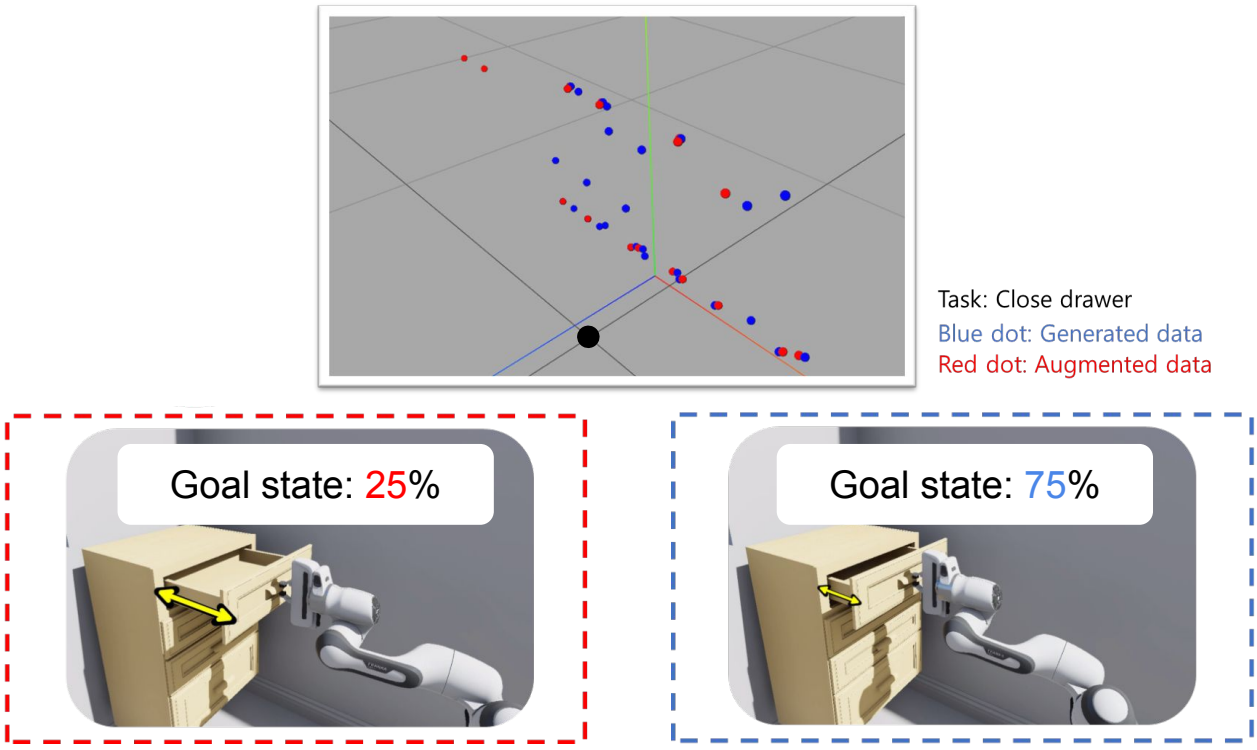


Figure 9. **Action augmentation for the *close drawer* task.** End-effector positions (pre-grasp, grasp, manipulation) from 10 dynamic scenes are visualized, with all robot base positions ($\boldsymbol{p}^{\text{base}}$) projected to the origin (black dots) for clarity. Blue and red dots represent generated and augmented actions evaluated by Isaac Sim for success. Simulation screens in blue and red dashed boxes illustrate a generated action (blue, goal state 75%) and an augmented action (red, new goal state 25%) from the same static scene.

Table 14. Ablation study results for DynScene with and without quaternion quantization and layout sampling.

	Eq. (6).	eq. (7).	P.OBJECT	R.OBJECT	O.DRAWER	C.DRAWER	O.CABINET	C.CABINET	P.WATER	T.WATER	AVERAGE
(a)	✗	✗	47.00	44.00	23.00	40.00	14.00	17.00	49.00	24.00	32.00
(b)	✗	✓	50.00	44.00	21.00	49.00	19.00	30.00	49.00	18.00	35.00
(c)	✓	✗	87.00	86.00	**49.00**	78.00	14.00	27.00	**85.00**	43.00	58.00
DynScene (Ours)	✓	✓	**92.00**	**88.00**	41.00	**95.00**	**37.00**	**58.00**	83.00	**62.00**	**69.50**

Table 15. **Task-wise diversity with top-k layout sampling.** For each task, 100 dynamic scenes were generated using layouts with the top k smallest distances between the robot base and object positions in the static scene.

Task	Top 1	Top 3	Top 5	Top 10
Pickup Object	1.00	1.01	1.07	**1.45**
Reorient Object	1.00	1.00	1.03	**1.21**
Open Drawer	1.00	1.00	1.01	**1.10**
Close Drawer	1.00	1.01	1.02	**1.11**
Open Cabinet	1.00	1.01	1.07	**1.26**
Close Cabinet	1.00	1.00	1.03	**1.18**
Pour Water	1.00	1.09	1.16	**1.82**
Transfer Water	1.00	1.12	1.17	**1.40**

D. Additional Ablation Studies

This section presents further ablation studies on Top-k layout sampling and the choice of quantization interval for quaternion quantization. These experiments help determine optimal settings for the model to balance diversity and success rates effectively.

Effect of Layout Sampling and Quaternion Quantization Table 14 presents the ablation study results for layout sampling (equation 6) and quaternion quantization (equation 7). Model (a) resulted in the lowest success rate of 32%, indicating that both components are crucial for the generation process. Model (b), which uses only layout sampling, shows limited improvement as objects still suffer from orientation

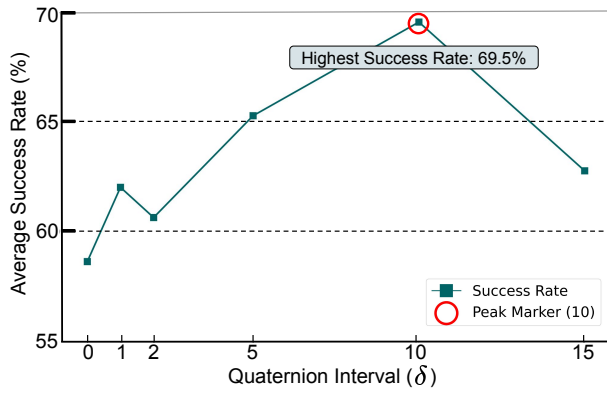


Figure 10. **Average success rates by quantization interval.** A value of $\delta = 10$ achieves the highest success rate of 69.50%, marking it as the optimal choice for performance.

issues during interaction. In contrast, Model (c), which applies only quaternion quantization, achieves slightly better results than Model (b) but struggles with accurate object placement. These results highlight that both components must work together to achieve best performance.

Top-k Layout Sampling The layout sampling method chooses the room layout with the smallest distance between static scene coordinates $\boldsymbol{p}^{\text{base}}$ (robot base) and $\boldsymbol{p}^{\text{obj}}$ (object). We evaluated its diversity by comparing Top 1 against Top k selections, where $k = 3, 5, 10$, using 100 dynamic scenes per task with k layouts each. Diversity is quantified as the average number of unique layouts, normalized to the Top 1 baseline (1.00). Table 15 shows that larger k values slightly increase diversity, with Top 10 achieving the highest across all tasks. However, Top 10 consistently lowers the success rate of dynamic scene generation, likely due to less optimal layouts. To balance diversity and success rate, we adopt the Top 1 layout with the smallest distance.

Quaternion Quantization with δ Quaternion quantization ensures precise orientation alignment between generated objects and the robot. Figure 10 presents the average success rates for quantization intervals (δ) in Equation 7: 0, 1, 2, 5, 10, and 15. As shown, $\delta = 10$ yields the highest success rate. Based on these results, DynScene adopts a quantization interval of 10.