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

Sangmin Lee[*], Sungyong Park[*], Heewon Kim
Soongsil University
{sm32289, ejqdl010}@gmail.com, hwkim@ssu.ac.kr

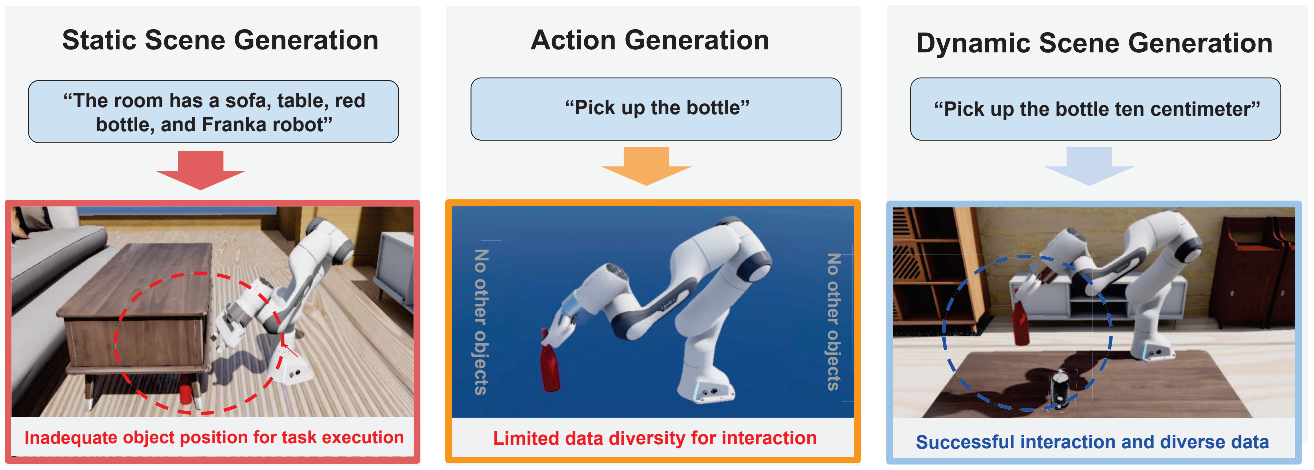

Static Scene Generation
"The room has a sofa, table, red bottle, and Franka robot"
Inadequate object position for task execution
Action Generation
"Pick up the bottle"
No other objects
No other objects
Limited data diversity for interaction
Dynamic Scene Generation
"Pick up the bottle ten centimeter"
Successful interaction and diverse data

Figure 1. **Types of text-guided data generation for embodied AI.** Static scene generation often prevents task completion due to object positioning, regardless of the task requirements (**left**). Action generation is limited by a lack of interaction diversity, failing to account for the effects of surrounding environments (**middle**). To the best of our knowledge, we are the first to propose a method for dynamic scene generation that enables more realistic and diverse task interactions by the joint generation of static scenes and robotic actions (**right**).

Abstract

Robotic manipulation in embodied AI critically depends on large-scale, high-quality datasets that reflect realistic object interactions and physical dynamics. However, existing data collection pipelines are often slow, expensive, and heavily reliant on manual efforts. We present DynScene, a diffusion-based framework for generating dynamic robotic manipulation scenes directly from textual instructions. Unlike prior methods that focus solely on static environments or isolated robot actions, DynScene decomposes the generation into two phases static scene synthesis and action trajectory generation allowing fine-grained control and diversity. Our model enhances realism and physical feasibility through scene refinement (layout sampling, quaternion quantization) and leverages residual action representation to enable action augmentation, generating multiple diverse trajectories from a single static configuration. Experiments show DynScene achieves 26.8× faster generation, 1.84× higher accuracy, and 28% greater action diversity than human-crafted data. Furthermore, agents trained with DynScene exhibit up to 19.4% higher success rates across complex manipulation tasks. Our approach paves the way for scalable, automated dataset generation in robot learning.

[*]These authors contributed equally.

1. Introduction

Embodied AI aims to train agents that can perceive, reason, and act within physically grounded environments, ultimately enabling robots to perform complex tasks in the real world. However, collecting real-world data for robotic manipulation is notoriously costly, time-consuming, and often unsafe for trial-and-error learning. To address this, the field has increasingly turned to simulation-based training [7, 10, 23, 30, 34, 39, 48], where agents can learn from interactions within virtual environments at scale.

This shift has led to the development of numerous simulation platforms such as AI2-THOR [13], Habitat 2.0 [34], and Isaac Sim [20], which offer physically plausible 3D environments for training embodied agents. Alongside simulators, manipulation-focused datasets like RLBench [10], ALFRED [30], and ARNOLD [7] have emerged to support research on tasks involving object interaction, spatial reasoning, and language grounding. These simulation datasets provide human-demonstrated or synthetic trajectories, allowing agents to learn from structured manipulation experiences.

Although these resources have significantly advanced the field, collecting and scaling such datasets remains a major challenge. To reduce manual effort, researchers have explored data generation techniques, including scene randomization, object rearrangement, and more recently, text-guided generative models for static scene synthesis [3, 35, 45] and robotic action generation [8, 18]. However, these two branches scene generation and action generation have largely evolved independently.

Critically, no prior work has addressed the joint generation of static scenes and corresponding robot actions in a coherent, physically grounded manner. Robot learning requires tightly coupled data for real-world deployment where object placements, spatial layouts, and robot motions are mutually consistent, dynamically plausible, and aligned with task goals. Achieving this demands high-fidelity scene synthesis and nuanced action modeling that accounts for physical constraints and interaction dynamics.

To fill this gap, we introduce DynScene, a diffusion-based framework that generates dynamic scenes comprising both a static scene and a sequence of robot actions from natural language instructions. A static scene defines the initial layout and poses of the robot and objects, while actions are represented as residual changes in the end-effector's position and orientation. DynScene incorporates (1) layout sampling to select collision-free room configurations, (2) quaternion quantization for object stability, (3) residual action representation for generalization, and (4) action augmentation to increase trajectory diversity from a single static setup. Crucially, all generated dynamic scenes are verified in a physics simulator, and only successful ones are used for training.

Our experiments demonstrate that DynScene significantly improves over human-generated and prior synthetic datasets, achieving 26.8× faster scene generation, 1.84× higher accuracy, and 28% more diverse actions. Moreover, when used to train manipulation agents, DynScene consistently boosts success rates up to 19.4% for PerAct and 10.03% for PerAct-PSA on the ARNOLD benchmark, especially in tasks requiring fine-grained control and continuous state changes.

In summary, our contributions are:

- A unified framework that generates dynamic robotic manipulation scenes including static environments and robot actions from textual instructions.
- A novel residual action formulation that enhances generalization across varied spatial configurations.
- A set of diffusion-based generation techniques that ensure physical plausibility and scene diversity, including layout sampling and quaternion quantization.
- Empirical evidence that DynScene enables more efficient and effective training of manipulation agents, outperforming existing baselines in both speed and task performance.

Table 1. **Scope of data generation methods for embodied AI.** While prior methods generate either static scenes or robot actions without considering physical interactions or continuous states, DynScene integrates both to produce coherent, diverse and physically grounded data.

Method	Data for Generation	Physical Conflict	Continuos State	Language
SynthER [18]	Robot Action	×	×	×
MTDIFF [8]	Robot Action	×	×	✓
ROSIE [45]	Image for Static Scene	×	×	✓
GenAug [3]	Image for Static Scene	×	×	✓
PhyScene [43]	Static Scene	✓	×	×
DiffuScene [35]	Static Scene	✓	×	✓
DynScene (Ours)	Static Scene & Robot Action	✓	✓	✓

2. Related Works

Simulators in Embodied AI Simulators are tools that represent environments in which robots operate, simulating various elements such as robot movements, object interactions, and sensor feedback in a virtual environment. Accurate modeling of real-world physical properties is crucial for developing sophisticated embodied AI systems. Recently, simulators have become increasingly specialized to meet specific research requirements and enhance the fidelity of simulations. AI2-THOR [13] provides realistic 3D indoor environments where agents can interact with objects to complete tasks, and is commonly used in Visual Question Answering (VQA). Habitat 2.0 [34] offers a high-performance physics engine and reconfigurable 3D apartment data, suited for training robots in diverse navigation and manipulation tasks, and supports research on long-term structured tasks. VirtualHome [26] enables agents to learn complex household tasks through programmatic representations, effectively advancing research in video understanding and task sequence learning. NVIDIA's Isaac Sim [20] provides advanced physics-based simulations capable of replicating intricate interactions, making it particularly valuable for studies on complex interaction dynamics. DynScene adopts Isaac Sim for realistic robot-object interactions and environments.

Datasets in Embodied AI In embodied AI, datasets have advanced to enable robots to perform and evaluate increasingly complex and sophisticated actions. Early stud-

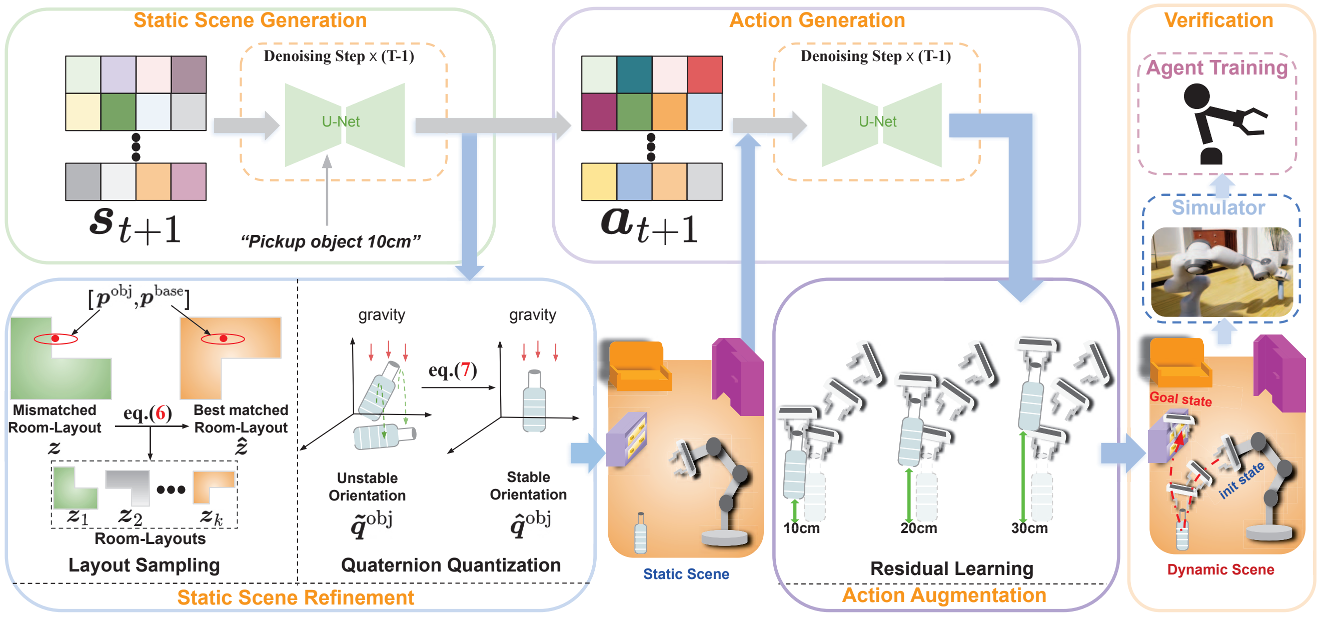


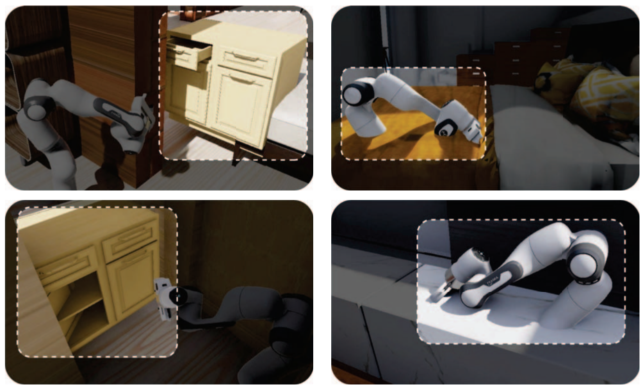
Figure 2. **Overview of the DynScene framework.** Using textual instruction, a diffusion model first generates the robot and target object for the static scene, while layout sampling determines additional elements, such as distractor objects and structural components. Quaternion quantization is applied to ensure the physical stability of the target object. Another diffusion model then generates robot actions from the static scene using residual coordinates, enabling data diversification through action augmentation. The resulting dynamic scenes are verified in simulation before being used for agent training.

ies [2, 48] focused on simple visual exploration tasks (*e.g.*, *move forward*), and subsequent research expanded to more complex visual navigation tasks (*e.g.*, *navigate to the room*), allowing robots to explore complex environments based on visual information. Recently, datasets incorporating interactions with objects (*e.g.*, *catch object*) [6, 7, 10, 14, 16, 19, 21–23, 30, 33, 34, 39, 40, 46] have emerged, enabling robots to recognize and explore objects in complex environments and learn to modify object states. Notable datasets include Gibson [39], used for indoor exploration and navigation learning in 3D environments. ALFRED [30] supports robots in performing complex tasks that involve manipulating objects and changing their states based on natural language instructions. This facilitates comprehensive visual perception, language understanding, and action planning evaluation. More recently, ARNOLD [7] provides environments for continuous state changes and high-resolution physical simulation, enabling robots to manipulate objects and learn goal-oriented behaviors in indoor settings. Collecting robotic manipulation data for complex tasks is labor-intensive, and scaling up to large datasets remains a significant challenge. DynScene aims to automate and scale up the generation of manipulation data to train agents for continuous states.

Data Generation for Embodied AI As the scale of embodied AI datasets has grown, the demand for efficient data generation methods has also increased. Early approaches addressed this by applying scene randomization techniques—modifying lighting, object materials, or swapping objects within the same class—to diversify RGB and depth observations [13, 39]. Later methods introduced scene augmentation strategies, such as altering object states or varying room layouts [29], and employed generative models for synthetic scene creation [15, 24, 25, 27, 28, 35–38, 41–43, 47]. For instance, ATISS [25] generates furniture layouts, while DiffuScene [35] and PhyScene [43] use diffusion-based models to construct physically plausible static environments with interactive objects. For action data, models like SynthER [18] and MTDIFF [8] generate robot trajectories using diffusion models for reinforcement learning. Other techniques, such as ROSIE [45] and GenAug [3], augment static scenes through text-guided image synthesis. However, these methods typically focus on either static scenes or robot actions in isolation and fail to ensure physical consistency across the entire manipulation sequence. As a result, human effort is still required to pair scenes with valid robot trajectories. In contrast, DynScene unifies static scene and action generation, ensuring physical plausibility and diversity and enabling scalable data synthesis. Table 1 summarizes the scope of data generation methods in embodied AI.

3. DynScene Framework

We present DynScene, a diffusion-based framework for generating dynamic scenes comprising a static environment and corresponding robot actions from natural language instructions. This section describes the full pipeline: data


(a) Improper room layout causing inaccessibility or collision

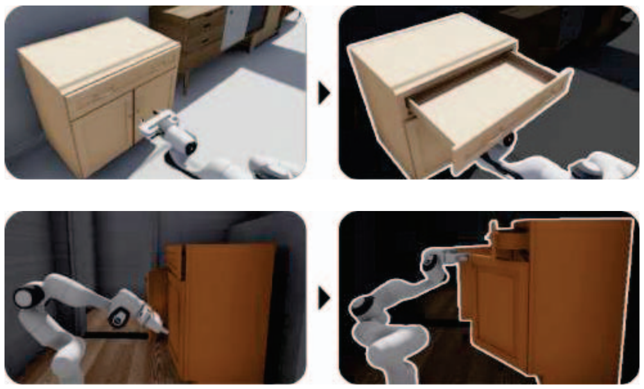

(b) Improper object orientation leading to instability

Figure 3. **Physical interaction issues due to inadequate static scene configuration:** (a) An improper room layout leads to objects and robots colliding with the environment, such as robots embedded within walls or objects submerged beneath floors or causing inaccessibility where the robot arm cannot reach the target object. (b) Improper object orientation causes the objects to shift due to gravity, altering the initial configuration of the target objects.

representation (Section 3.1), static scene generation with physics-based refinement (Section 3.2), robot action generation and augmentation (Section 3.3), and agent training using only simulation-verified samples (Section 3.4). Figure 2 illustrates the overview of the DynScene framework.

3.1. Data Representation for Dynamic Scene

We define a dynamic scene as a pair $(\boldsymbol{s}, \boldsymbol{a})$, where $\boldsymbol{s}$ is a static scene representing the initial configuration of the robot and objects, and $\boldsymbol{a}$ is a sequence of robot actions driven by a task instruction y (*e.g.*, "close the drawer by 50 percent"). This structured representation enables the generation of diverse and coherent combinations of environments and manipulation behaviors.

Static Scene Data using Absolute Coordinates The static scene data ($\boldsymbol{s}$) is a set of information about the target objects ($\boldsymbol{o}$), robot ($\boldsymbol{r}$), and room layouts ($\boldsymbol{z}$) as following equation 1:

$$\boldsymbol{s} = [\boldsymbol{o}, \boldsymbol{r}, \boldsymbol{z}], \tag{1}$$

where $\boldsymbol{o}$ is $[\boldsymbol{p}^{\text{obj}}, \boldsymbol{q}^{\text{obj}}, \boldsymbol{c}, \boldsymbol{v}, \boldsymbol{f}, \boldsymbol{s}^{\text{init}}, \boldsymbol{s}^{\text{goal}}]$ and $\boldsymbol{p}^{\text{obj}} \in \mathbb{R}^3$ and $\boldsymbol{q}^{\text{obj}} \in \mathbb{R}^4$ are the position and quaternion orientation of the object. The object class $\boldsymbol{c} \in \mathbb{R}^C$ (*e.g., 'cup', 'drawer'*) indicates one of C classes, and $\boldsymbol{v} \in \mathbb{R}^3$ represents the object size. The shape code $\boldsymbol{f} \in \mathbb{R}^{32}$ is a point cloud embedding vector. Additionally, $\boldsymbol{s}^{\text{init}}$ and $\boldsymbol{s}^{\text{goal}} \in \mathbb{R}$ denote the initial and goal states of the task. $\boldsymbol{r} = [\boldsymbol{p}^{\text{base}}, \boldsymbol{q}^{\text{base}}, \boldsymbol{p}_0^{\text{ee}}, \boldsymbol{q}_0^{\text{ee}}, \boldsymbol{g}_0]$ includes the robot's base position and quaternion orientation, denoted as $\boldsymbol{p}^{\text{base}} \in \mathbb{R}^3$ and $\boldsymbol{q}^{\text{base}} \in \mathbb{R}^4$. The end-effector's position and quaternion orientation are represented by $\boldsymbol{p}_0^{\text{ee}}$ and $\boldsymbol{q}_0^{\text{ee}}$, where $\boldsymbol{g}_0$ indicates the gripper state (*e.g., 'open', 'close'*). Finally, $\boldsymbol{z}$ denotes the room layout, which defines the configuration of environments, including distractor objects and structural elements of the room.

Action Data using Residual Coordinates The robot actions represent the end-effector's motion, divided into K steps tailored to each task, where K is a task-specific hyperparameter. The action data for the k-th step ($\boldsymbol{a}_k$) is a set of residual positions ($\Delta\boldsymbol{p}_k^{\text{ee}}$), residual quaternion orientations ($\Delta\boldsymbol{q}_k^{\text{ee}}$), and gripper states ($\boldsymbol{g}_k$) as in equation 2:

$$\boldsymbol{a}_k = [\Delta\boldsymbol{p}_k^{\text{ee}}, \Delta\boldsymbol{q}_k^{\text{ee}}, \boldsymbol{g}_k], \tag{2}$$

where $\Delta\boldsymbol{p}_k^{\text{ee}} = \boldsymbol{p}_k^{\text{ee}} - \boldsymbol{p}_{k-1}^{\text{ee}}$ and $\Delta\boldsymbol{q}_k^{\text{ee}} = \boldsymbol{q}_k^{\text{ee}} - \boldsymbol{q}_{k-1}^{\text{ee}}$ for $k = 1, 2, \ldots, K$. This residual approach ensures consistent learning across identical tasks, independent of static scenes, and is applicable to any static scene for action augmentation.

3.2. Static Scene Generation and Refinement

Conditional Diffusion with Instruction We employ a conditional diffusion model to generate static scene ($\boldsymbol{s}$) from a text-based manipulation instruction (y), such as *"close the cabinet to 30 percent."* The forward process, described in equation 3, gradually adds Gaussian noise to static scene data, represented by $\boldsymbol{s}_0$, transforming it into pure noise $\boldsymbol{s}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ over T timesteps:

$$q(\boldsymbol{s}_t \mid \boldsymbol{s}_0) = \mathcal{N}\left(\boldsymbol{s}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{s}_0, (1-\bar{\alpha}_t)\mathbf{I}\right). \tag{3}$$

The reverse process in equation 4 describes iterative denoising from $\boldsymbol{s}_T$ to reconstruct $\boldsymbol{s}_0$ using the text condition (y):

$$p_\theta(\boldsymbol{s}_{t-1} \mid \boldsymbol{s}_t, y) = \mathcal{N}\left(\boldsymbol{s}_{t-1}; \mu_\theta(\boldsymbol{s}_t, t; y), \sigma_t^2\mathbf{I}\right). \tag{4}$$

The training objective is to estimate the noise ϵ added during the forward process by minimizing the following equation 5,

$$\mathcal{L}(\theta)_{\text{text}} := \mathbb{E}_{\boldsymbol{s}_0,\epsilon,t}\left[\|\epsilon - \epsilon_\theta(\boldsymbol{s}_t, t; y)\|^2\right], \tag{5}$$

where $\boldsymbol{s}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{s}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$. The denoising network $\epsilon_\theta(\cdot)$ uses a U-Net architecture [35], incorporating instruction embeddings into attention layers using a pre-trained language model [11]. This diffusion model effectively generates realistic static scenes that align with the specified instructions. This process omits the room layout $\boldsymbol{z}$ in $\boldsymbol{s}$ and determines it through sampling, as described in the following section.

Layout Sampling using Position Differences While the generation of room layouts has recently received increased attention [25, 29, 35, 43], our interest lies in modeling the actions between the robot and the target objects for given layouts. As shown in Figure 3a, an improper room layout can lead to a collision between the robot and objects or make the robot arm unable to reach the target object, making the task impossible to complete. To address such layout issues, we utilize a layout sampling method from given datasets that can be handcrafted or generated, as described in equation 6:

$$\begin{aligned}\hat{r} &= \arg\min_r(\|\boldsymbol{p}_r^{\text{obj}} - \tilde{\boldsymbol{p}}^{\text{obj}}\|^2 + \|\boldsymbol{p}_r^{\text{base}} - \tilde{\boldsymbol{p}}^{\text{base}}\|^2) \\ \hat{\boldsymbol{z}} &= \boldsymbol{z}_{\hat{r}},\end{aligned} \tag{6}$$

where $\hat{\boldsymbol{z}}$ is the sampled layout, $\boldsymbol{z}_r$ is the r-th room layout in the dataset, $\boldsymbol{p}_r^{\text{obj}}$ and $\boldsymbol{p}_r^{\text{base}}$ are the object and robot base positions in the r-th layout, and $\tilde{\boldsymbol{p}}^{\text{obj}}$ and $\tilde{\boldsymbol{p}}^{\text{base}}$ are the corresponding predictions from the diffusion model.

Quaternion Quantization to Prevent Object Collapse Gravity forces objects to tilt and collapse due to small changes (or errors from prediction models) in their orientation. This phenomenon is particularly problematic for articulated objects, such as cabinets and drawers, given that the objects move before robot actions and hinder interactions, as illustrated in Figure 3b. To alleviate this problem, the predicted object orientation ($\tilde{\boldsymbol{q}}^{\text{obj}}$) from the diffusion model is quantized through a fixed interval δ as equation 7:

$$\hat{\boldsymbol{q}}^{\text{obj}} = \text{round}\left(\frac{\tilde{\boldsymbol{q}}^{\text{obj}}}{\delta}\right) \cdot \delta. \tag{7}$$

3.3. Action Generation and Augmentation

Conditional Diffusion with Static Scene To generate the action data from a static scene, we recall the diffusion model in Section 3.2 and modify its denoised and conditioned data accordingly. The forward process, the reverse process, and the objective for action generation become equation 8:

$$\begin{aligned}q(\boldsymbol{a}_t \mid \boldsymbol{a}_0) &= \mathcal{N}\left(\boldsymbol{a}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{a}_0, (1-\bar{\alpha}_t)\mathbf{I}\right) \\ p_\phi(\boldsymbol{a}_{t-1} \mid \boldsymbol{a}_t, \mathbf{s}) &= \mathcal{N}(\boldsymbol{a}_{t-1}; \mu_\phi(\boldsymbol{a}_t, t; \mathbf{s}), \sigma_t^2\mathbf{I}) \\ \mathcal{L}(\phi)_{\text{scene}} &:= \mathbb{E}_{\boldsymbol{a}_0,\epsilon,t}[\|\epsilon - \epsilon_\phi(\boldsymbol{a}_t, t; \mathbf{s})\|^2],\end{aligned} \tag{8}$$

where t is the timestep in the the diffusion process and $\boldsymbol{a}_0$ represents the complete set of K action steps ($[\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_K]$), with the subscript 0 referring to the initial diffusion timestep. $\boldsymbol{a}_T$ is Gaussian noise, $\epsilon_\phi(\cdot)$ is the denoising network, and $\boldsymbol{s}$ is the static scene data.

Action Augmentation By employing residual learning through residual coordinates and separating data into static scenes and actions, our model effectively generates diverse, task-specific actions for a wide range of static scenes. For example, by creating 10 static scenes and generating 10 corresponding actions for each, we can augment the dataset to 100 (10×10) diverse actions. Moreover, since actions are trained to generate actions based on specific tasks and states, actions can be augmented for states not originally present in the static scene by merging actions learned from different states. For instance, in the *pickup object* task, if the static scene only includes a 10cm setup, the actions can generate actions for other states, such as 20cm or 30cm, by training on static scenes with these different states. This method enables the augmentation of diverse actions, facilitating efficient data generation and allowing the robot to learn a variety of goal states beyond simple augmentations. More detailed information can be found in the appendix.

Filtering Invalid Dynamic Scenes Generated dynamic scenes are used to train agents for robotic manipulation tasks. However, inaccurate or unnecessary actions can lead to task failures and degrade agent performance. We use Isaac Sim [20] to verify that the generated actions are executable with the success threshold from prior work [7] to filter out invalid dynamic scenes. As a result, agents are trained only on scenes where tasks are completed, preventing inaccurate actions from negatively impacting the learning process.

3.4. Phase-Aware Agent Training

To create effective agents for robotic manipulation tasks, we employed PerAct [31], a baseline model that achieved high performance in previous research [7]. Additionally, we modified PerAct by training the grasping and manipulation phases using separate networks. By dividing the training in this way, each network can specialize in its specific phase, enabling more refined and precise actions, especially in complex manipulation tasks. This approach, denoted PerAct-PSA, enhances the model's ability to learn phase-specific features, thereby improving overall manipulation performance.

4. Experiments

4.1. Implementation Details

Datasets We utilize the ARNOLD [7] benchmark for training and evaluation of DynScene. This benchmark includes 8 manipulation tasks for robot learning through language instructions and provides 3,571 training samples with 773 test samples across these tasks. Each task offers continuous object states (*e.g., "open the drawer 75%"* or *"pour half the water out"*) in realistic 3D environments, unlike other benchmarks that typically focus on discrete states.

Table 2. **Quantitative comparison on unconditional sequential action generation.** We compare ARNOLD and DynScene on diversity metrics: Fréchet Distance (FD), Dynamic Time Warping (DTW), and Spatial Coverage (SC) across various robotic manipulation tasks. Note that a factor of 10^3 scales SC values for comparability with other metrics.

Task	Fréchet Distance (FD) ↑		Dynamic Time Warping (DTW) ↑		Spatial Coverage (SC) ↑	
	ARNOLD [7]	**DynScene (Ours)**	ARNOLD [7]	**DynScene (Ours)**	ARNOLD [7]	**DynScene (Ours)**
Pickup Object	32.85	**36.38**	54.77	**61.23**	2.94	**3.29**
Reorient Object	23.49	**27.45**	27.98	**35.56**	2.30	**2.73**
Close Cabinet	37.77	**43.85**	76.72	**86.01**	**4.30**	3.21
Open Cabinet	**40.83**	40.59	81.71	**83.54**	2.92	**3.17**
Close Drawer	24.85	**25.55**	35.47	**37.86**	1.66	**3.78**
Open Drawer	**28.36**	24.36	**44.98**	39.51	2.67	**4.04**
Pour Water	20.46	**26.52**	31.09	**61.62**	2.96	**4.91**
Transfer Water	**18.52**	17.84	27.90	**36.65**	2.89	**3.85**
Average	28.39	**30.32**	47.58	**55.25**	2.83	**3.62**

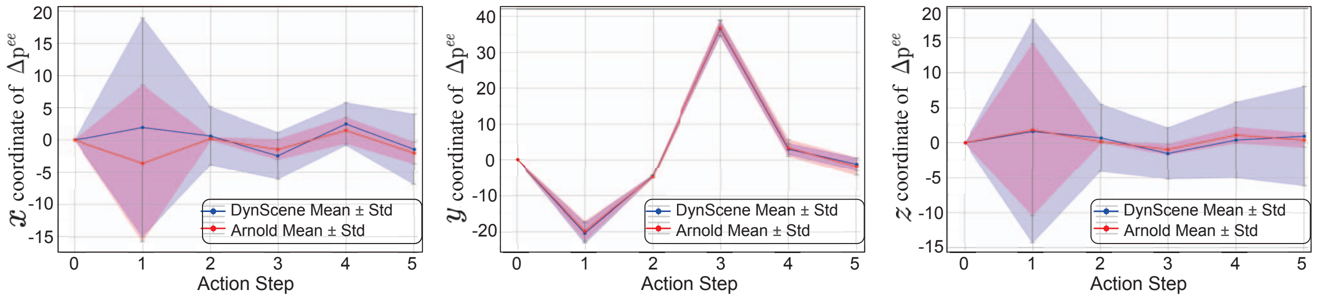


Figure 4. **Generated action diversity.** Mean and standard deviation of end-effector position changes ($\Delta \boldsymbol{p}_k^{\text{ee}}$) in the *'pour water'* task along the x, y, and z axes. DynScene exhibits wider x- and z-axis distributions than ARNOLD [7], indicating greater diversity, while y-axis movement remains similar due to task-specific vertical precision.

Table 3. **Task-wise dynamic scene generation time and success rate (%).**

Method	**Task**	**Time (sec)**	**Success Rate (%)**
DynScene (Ours)	P.OBJECT	2.53 ± 0.02	92.00
	R.OBJECT	2.52 ± 0.02	88.00
	C.CABINET	2.50 ± 0.02	58.00
	O.CABINET	2.53 ± 0.02	37.00
	C.DRAWER	2.52 ± 0.02	95.00
	O.DRAWER	2.52 ± 0.02	41.00
	P.WATER	2.52 ± 0.02	83.00
	T.WATER	2.52 ± 0.02	62.00
	Average	**2.52** ± 0.02	**69.50**
ARNOLD† [7]	Average	67.50	37.50
Efficiency (*vs.* ARNOLD†)		**26.8× faster**	**1.84× higher**

† robotic manipulation data collected by human experts.

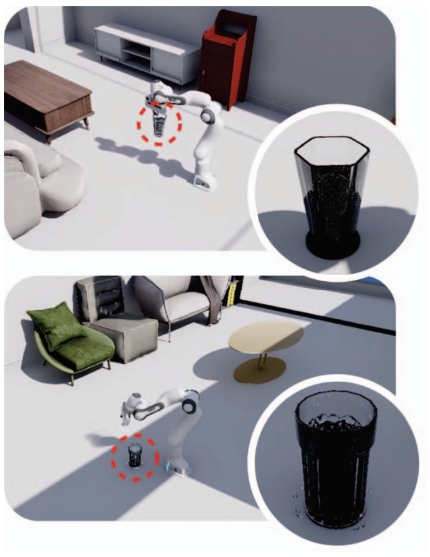
Diversity Metrics To evaluate the diversity of generated actions, we employ three metrics widely used in robot navigation research [5, 9, 17, 32]. Fréchet Distance (FD) [1] measures how varied the generated action paths are across different sequences. The temporal diversity of generated paths is evaluated by measuring path similarity using Dynamic Time Warping (DTW) [9]. The spatial diversity of robot positions is measured using Spatial Coverage (SC) [4], assessing the spatial spread of interactions across the scene.

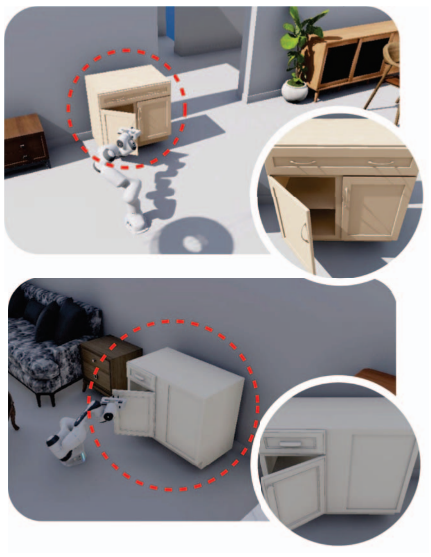
Success Rate (SR) SR evaluates the performance of dynamic scene generation and agents in robotic manipulation tasks. For dynamic scene generation, SR reports the average score from a single trial across 100 generated scenes for each task. For agents, SR reports the average score based on three trials using the test dataset unless otherwise specified.

Training Setting DynScene was trained for 100,000 epochs per task using the Adam [12] optimizer with a learning rate of 2e-4. For agent training, we conducted 200,000 training steps with a batch size of 4 and a learning rate of 1e-4 using LAMB [44] optimizer. Note that we only used scenes that successfully passed verification in Isaac Sim [20] using ARNOLD's [7] evaluation thresholds for agent training. We conducted our experiments for DynScene and agent using NVIDIA RTX4090 and A6000 GPUs, respectively.

4.2. Unconditioned Dynamic Scene Generation

To evaluate our method without biases from specific text prompts, we generated 100 dynamic scenes for each task in an unconditioned setting. We compared the efficiency and quality of generated scenes against human demonstrations [7] and measured the diversity of generated actions.


(a) “dump fifty percent of the water from the glass”


(b) “adjust the cabinet seventy-five percent closed”

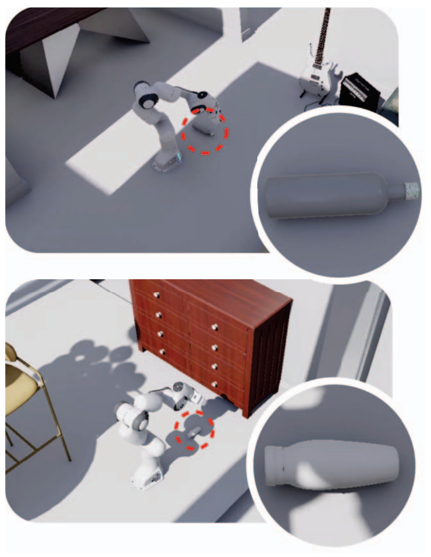

(c) “angle the bottle 45 degrees away from the high axis”

Figure 5. **Text-conditioned generated scenes.** Visualization of various scenes and objects generated with the same text prompt

Table 4. **Comparison to DiffuScene [35] (static scene generation).**

Static Scene	Robot Action	Object Entropy↑	Scene Entropy↑	Generation Time↓	Success Rate↑
DiffuScene	ARNOLD	**0.94**	**0.93**	30.24 ± 0.28	20.04
DynScene	**DynScene**	0.93	0.91	**2.95** ± 0.21	**69.50**

Comparison with Human Expert Table 3 presents generation time and success rates (SR) between DynScene and the ARNOLD dataset [7]. Human experts using an Xbox controller to collect ARNOLD data in a 3D simulation, required an average of 67.50 seconds per scene. In contrast, DynScene achieved an average SR of 69.50%, outperforming the human experts of 37.50%. Moreover, DynScene generated scenes significantly faster, requiring only 2.52 seconds per scene approximately 26 times faster than human experts. These results highlight the efficiency of our approach in generating training data for robotic manipulation tasks.

Comparison with Static Scene Generation Table 4 compares DynScene to DiffuScene [35], which is a specialized model for static scene generation, with action data from ARNOLD [7]. DiffuScene achieves slightly better entropy in the unconditioned object and scene generation. In contrast, DynScene achieves a 10× faster in generation time and 3.5× higher success rates. These results highlight the limitations of pairing static scenes with external actions and the effectiveness of our joint generation approach.

Comparison of Action Diversity Table 2 compares the diversity of dynamic scenes generated by DynScene with ARNOLD training data [7], using the same number of valid dynamic scenes per task. DynScene outperforms ARNOLD across all metrics: FD at 30.32 *vs.* 28.39 for varied action paths, DTW at 55.25 *vs.* 47.58 for temporal variability, and SC at 3.62 *vs.* 2.83 for broader gripper exploration. This result indicates that DynScene produces more diverse and spatially expansive actions than the training data.

Analysis of Action Diversity To explore the diversity of actions generated by DynScene, we analyzed variations in end-effector position changes ($\Delta \boldsymbol{p}_k^{\text{ee}}$) along the x, y, and z coordinates in the *‘pour water’* task, as shown in Figure 4. Compared to the ARNOLD dataset [7], DynScene exhibits notably wider distributions in Δx and Δz, indicating greater spatial diversity in action patterns. Conversely, the Δy distribution aligns closely with ARNOLD, a result of the task’s critical need for precise vertical positioning to pour water effectively. These findings highlight DynScene’s ability to enhance action diversity while respecting task-specific constraints. Please refer to the appendix for action diversity analyses across all tasks.

4.3. Text-Conditioned Scene Generation

Qualitative Results Figure 5 illustrates DynScene’s ability to generate diverse and dynamic scenes from identical text prompts. The outputs vary in object shapes, initial states (e.g., water levels), room layouts, and spatial arrangements of the robot and objects while maintaining physical plausibility and semantic alignment with the instruction. These results demonstrate that DynScene can produce various scene compositions without compromising task-relevant interactions.

Table 5. **Evaluation results of the agents on various tasks.** Success rate using the ARNOLD test dataset. For each agent, the first row shows performance trained on the original ARNOLD dataset, followed by a row for training on combined ARNOLD + DynScene scenes. Task names are abbreviated for brevity, and each row includes average performance across the 8 tasks. PerAct (ARNOLD) results are from the ARNOLD [7] paper.

Method	P.OBJECT	R.OBJECT	O.DRAWER	C.DRAWER	O.CABINET	C.CABINET	P.WATER	T.WATER	AVERAGE
BC-Lang-CNN (ARNOLD)	**5.00**	0.00	0.00	20.00	0.00	10.00	0.00	0.00	4.35
BC-Lang-CNN (DynScene + ARNOLD)	**5.00**	0.00	0.00	**25.00**	0.00	**20.00**	0.00	0.00	**6.20**
BC-Lang-ViT (ARNOLD)	1.67	0.00	0.00	35.00	0.00	10.00	0.00	0.00	5.84
BC-Lang-ViT (DynScene + ARNOLD)	**5.00**	0.00	0.00	**45.00**	0.00	**33.33**	0.00	0.00	**10.42**
PerAct (ARNOLD)	88.81	3.90	26.05	**33.78**	11.59	20.39	**34.33**	14.29	29.14
PerAct (DynScene + ARNOLD)	**90.00**	**20.00**	**38.33**	30.00	**16.67**	**31.67**	30.00	**21.67**	**34.79**
PerAct-PSA (ARNOLD)	90.00	**30.00**	41.67	**51.67**	20.00	15.00	**63.33**	20.00	41.46
PerAct-PSA (DynScene + ARNOLD)	**95.00**	25.00	**43.33**	43.33	**45.00**	**38.33**	46.67	**28.33**	**45.62**

Table 6. **Goal state estimation and success rates of DynScene.** Each score reports the average of 10 generated trajectories.

Textual Prompt	**Goal State**	**Success Rate**
"increase the height of the bottle **ten** centimeters above the ground"	**9.99 ± 0.01**	100.00
"tilt the bottle about **180** degrees away from the upward axis"	**179.94 ± 0.01**	100.00
"open the cabinet a **quarter**"	**0.25 ± 0.00**	40.00
"put **eighty** percent of the liquid to the cup"	**80.09 ± 0.01**	80.00

Table 7. **Success rate on unseen objects, scenes, and states.** Better scores indicate better generalization. Any state refers to an intermediate state between seen states, while novel state lies outside the range of seen states.

Method	**Novel Object**	**Novel Scene**	**Novel State**	**Any State**
PerAct (AR)	14.53	19.00	**4.58**	11.01
PerAct (DS + AR)	**17.29**	**22.92**	3.30	**13.54**

AR: ARNOLD [7], DS: DynScene

Additional text prompts are provided in the appendix.

Quantitative Results Table 6 presents the success rates of dynamic scenes generated from text prompts, with each score averaged over 10 trajectories. DynScene accurately predicts numerical goal states, closely matching the values specified in the input prompts (*e.g.*, 9.99cm, 179.94°, and 80.09%). However, the success rate varies across tasks, which reflects differences in task difficulty. For instance, achieving a precise cabinet opening (40.00%) is more error-prone than an object's height or tilt adjustments (100.00%). This trend is consistent with the agent performance shown in Table 5, where tasks with lower success rates in scene generation result in lower agent success rates.

4.4. Evaluation with Manipulation Agents

Agent Performance Table 5 compares the success rates of agents trained solely on the ARNOLD dataset [7] versus those trained on a combination of ARNOLD and the data generated by DynScene. Incorporating DynScene significantly improves performance, particularly on challenging tasks such as *open cabinet* and *transfer water*, where models trained only on ARNOLD data showed limited success (average $\sim$30%). Notably, PerAct-PSA trained with DynScene outperformed the baseline across all tasks, highlighting the effectiveness of our approach for robotic manipulation.

Generalization Ability ARNOLD [7] includes unseen objects, scenes, and states to assess generalization ability in the test set. An any state refers to an intermediate state between seen states, while a novel state denotes a state that lies outside the range of observed states. Table 7 demonstrates that the data generated from DynScene enhances generalization, except for novel states, which can be considered out-of-distribution. For instance, training on pouring 25-75% of water while testing at 100%. Future work can develop more diverse state representations to handle these extreme out-of-distribution test scenarios.

5. Limitation

While DynScene shows strong performance on the ARNOLD [7] benchmark, it has only been evaluated on a single dataset. Generalization to other domains and more complex, out-of-distribution states remains an open challenge. Future work can explore broader datasets and richer state variations to improve robustness.

6. Conclusion

This paper introduces DynScene, a framework that automates the generation of dynamic scenes for robotic manipulation tasks in embodied AI. DynScene separately generates static scenes and robot actions, conditioning action generation on the static scene to account for physical interactions. This approach enables diverse action generation and allows for effective augmentation. Our experiments demonstrated that DynScene efficiently generates diverse and realistic scenes compared to human-collected datasets, significantly improving agent training and resulting in higher success rates. Future work will extend the framework to a broader range of tasks and further enhance the physical realism of the generated scenes.

Acknowledgements

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the National Program for Excellence in SW) (2024-0-00071), the Graduate School of Metaverse Convergence support program(IITP-2025-RS-2024-00430997), and the Convergence security core talent training business support program(IITP-2025-RS-2024-00426853) supervised by the IITP(Institute of Information & communications Technology Planning & evaluation). This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIT)(RS-2024-00456589). We thank Corca AI for sponsoring the compute used in this work.

References

[1] Helmut Alt and Michael Godau. Computing the fréchet distance between two polygonal curves. *IJCGA*, 5(01n02):75–91, 1995. 6

[2] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *CVPR*, 2020. 3

[3] Zoey Chen, Sho Kiami, Abhishek Gupta, and Vikash Kumar. Genaug: Retargeting behaviors to unseen situations via generative augmentation. *arXiv:2302.06671*, 2023. 2, 3

[4] Howie Choset. Coverage for robotics–a survey of recent results. *Annals of mathematics and artificial intelligence*, 31: 113–126, 2001. 6

[5] Jorge de Heuvel, Nathan Corral, Benedikt Kreis, Jacobus Conradi, Anne Driemel, and Maren Bennewitz. Learning depth vision-based personalized robot navigation from dynamic demonstrations in virtual reality. In *IROS*, 2023. 6

[6] Chuang Gan, Jeremy Schwartz, Seth Alter, Damian Mrowca, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, et al. Threedworld: A platform for interactive multi-modal physical simulation. *arXiv:2007.04954*, 2020. 3

[7] Ran Gong, Jiangyong Huang, Yizhou Zhao, Haoran Geng, Xiaofeng Gao, Qingyang Wu, Wensi Ai, Ziheng Zhou, Demetri Terzopoulos, Song-Chun Zhu, et al. Arnold: A benchmark for language-grounded task learning with continuous states in realistic 3d scenes. In *ICCV*, 2023. 1, 2, 3, 5, 6, 7, 8

[8] Haoran He, Chenjia Bai, Kang Xu, Zhuoran Yang, Weinan Zhang, Dong Wang, Bin Zhao, and Xuelong Li. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. *NeurIPS*, 2023. 2, 3

[9] Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. General evaluation for instruction conditioned navigation using dynamic time warping. *arXiv:1907.05446*, 2019. 6

[10] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *RA-L*, 5(2):3019–3026, 2020. 1, 2, 3

[11] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019. 5

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014. 6

[13] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv:1712.05474*, 2017. 2, 3

[14] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *CoRL*, 2023. 3

[15] Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. Grains: Generative recursive autoencoders for indoor scenes. *TOG*, 38(2):1–16, 2019. 3

[16] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *CoRL*, 2021. 3

[17] Bo Ling, Yan Lyu, Dongxiao Li, Guanyu Gao, Yi Shi, Xueyong Xu, and Weiwei Wu. Socialgail: Faithful crowd simulation for social robot navigation. In *ICRA*, 2024. 6

[18] Cong Lu, Philip Ball, Yee Whye Teh, and Jack Parker-Holder. Synthetic experience replay. *NeurIPS*, 2023. 2, 3

[19] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. *arXiv:2005.07648*, 2020. 3

[20] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv:2108.10470*, 2021. 2, 5, 6

[21] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *RA-L*, 7(3):7327–7334, 2022. 3

[22] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, et al. Orbit: A unified simulation framework for interactive robot learning environments. *RA-L*, 8(6):3740–3747, 2023.

[23] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv:2107.14483*, 2021. 1, 3

[24] Yinyu Nie, Angela Dai, Xiaoguang Han, and Matthias Nießner. Learning 3d scene priors with 2d supervision. In *CVPR*, 2023. 3

[25] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. *NeurIPS*, 2021. 3, 5

[26] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *CVPR*, 2018. 2

[27] Pulak Purkait, Christopher Zach, and Ian Reid. Sg-vae: Scene grammar variational autoencoder to generate new indoor scenes. In *ECCV*, 2020. 3

[28] Daniel Ritchie, Kai Wang, and Yu-an Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *CVPR*, 2019. 3

[29] Hongrui Sang, Rong Jiang, Zhipeng Wang, Yanmin Zhou, Ping Lu, and Bin He. Scene augmentation methods for interactive embodied ai tasks. *TIM*, 72:1–11, 2023. 3, 5

[30] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *CVPR*, 2020. 1, 2, 3

[31] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *CoRL*, 2022. 5

[32] Daeun Song, Jing Liang, Xuesu Xiao, and Dinesh Manocha. Tgs: Trajectory generation and selection using vision language models in mapless outdoor environments. *arXiv:2408.02454*, 2024. 6

[33] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, Karen Liu, et al. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *CoRL*, 2022. 3

[34] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *NeurIPS*, 2021. 1, 2, 3

[35] Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In *CVPR*, 2024. 2, 3, 4, 5, 7

[36] Kai Wang, Manolis Savva, Angel X Chang, and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. *TOG*, 37(4):1–14, 2018.

[37] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. In *3DV*, 2021.

[38] Mingdong Wu, Fangwei Zhong, Yulong Xia, and Hao Dong. Targf: Learning target gradient field to rearrange objects without explicit goal specification. *NeurIPS*, 2022. 3

[39] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, 2018. 1, 3

[40] Eliot Xing, Abhinav Gupta, Sam Powers, and Victoria Dean. Kitchenshift: Evaluating zero-shot generalization of imitation-based policy learning under domain shifts. In *NeruIPS*, 2021. 3

[41] Haitao Yang, Zaiwei Zhang, Siming Yan, Haibin Huang, Chongyang Ma, Yi Zheng, Chandrajit Bajaj, and Qixing Huang. Scene synthesis via uncertainty-driven attribute synchronization. In *ICCV*, 2021. 3

[42] Ming-Jia Yang, Yu-Xiao Guo, Bin Zhou, and Xin Tong. Indoor scene generation from a collection of semantic-segmented depth images. In *ICCV*, 2021.

[43] Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. Physcene: Physically interactable 3d scene synthesis for embodied ai. In *CVPR*, 2024. 2, 3, 5

[44] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv:1904.00962*, 2019. 6

[45] Tianhe Yu, Ted Xiao, Austin Stone, Jonathan Tompson, Anthony Brohan, Su Wang, Jaspiar Singh, Clayton Tan, Jodilyn Peralta, Brian Ichter, et al. Scaling robot learning with semantically imagined experience. *arXiv:2302.11550*, 2023. 2, 3

[46] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. In *CoRL*, 2021. 3

[47] Zaiwei Zhang, Zhenpei Yang, Chongyang Ma, Linjie Luo, Alexander Huth, Etienne Vouga, and Qixing Huang. Deep generative modeling for scene synthesis via hybrid representations. *TOG*, 39(2):1–21, 2020. 3

[48] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 2017. 1, 3