ClutterGen: A Cluttered Scene Generator for Robot Learning

Yinsen Jia **Boyuan Chen**
Duke University
http://generalroboticslab.com/ClutterGen

Abstract: We introduce **ClutterGen**, a physically compliant simulation scene generator capable of producing highly diverse, cluttered, and stable scenes for robot learning. Generating such scenes is challenging as each object must adhere to physical laws like gravity and collision. As the number of objects increases, finding valid poses becomes more difficult, necessitating significant human engineering effort, which limits the diversity of the scenes. To overcome these challenges, we propose a reinforcement learning method that can be trained with physics-based reward signals provided by the simulator. Our experiments demonstrate that ClutterGen can generate cluttered object layouts with up to ten objects on confined table surfaces. Additionally, our policy design explicitly encourages the diversity of the generated scenes for open-ended generation. Our real-world robot results show that ClutterGen can be directly used for clutter rearrangement and stable placement policy training.

Keywords: Simulation Scene Generation, Manipulation, Robot Learning

Fig. 1: **(a) The success rate of generating a stable simulation setup.** When the number of objects in the environment increases, the difficulty of creating such a stable setup also increases. The traditional heuristic method cannot create a simulation scene above 7 objects, while ClutterGen consistently achieves high success rates. **(b) Diverse, cluttered, and stable simulation setups created by ClutterGen.**

1 Introduction

Simulation has played an important role in advancing robot learning [1, 2, 3, 4, 5]. Significant advancements in robot learning have been achieved by randomizing object shapes [4, 6, 7], textures [8, 9, 10, 11], and dynamics [12]. Unlike object properties, which can be easily specified within a range without interfering with other objects, object layout must consider the presence of other objects and physical feasibility. For instance, arranging objects in a scene requires ensuring that they do not overlap and are placed in stable positions instead of falling down from the air. Existing efforts often prevent this issue by fixing the object bases [13, 4, 14, 15], but this strategy is not suitable for many movable objects like bottles or cups. As the number of objects increases within a limited space, generating a randomized yet stable object layout becomes exponentially difficult. Fig. 1(a) shows the challenge of using the widely adopted approach of random sampling and rejecting failure trials [16, 17, 18, 19] to generate valid scenes, with even seven objects on a table. Other methods require human manual specifications of object regions for local randomization [20, 21, 22] or apply discretization to the possible placement space to avoid collisions [17, 23, 24]. However, navigating and manipulating cluttered environments are essential challenges to deploying robot learning to the real world.

8th Conference on Robot Learning (CoRL 2024), Munich, Germany.

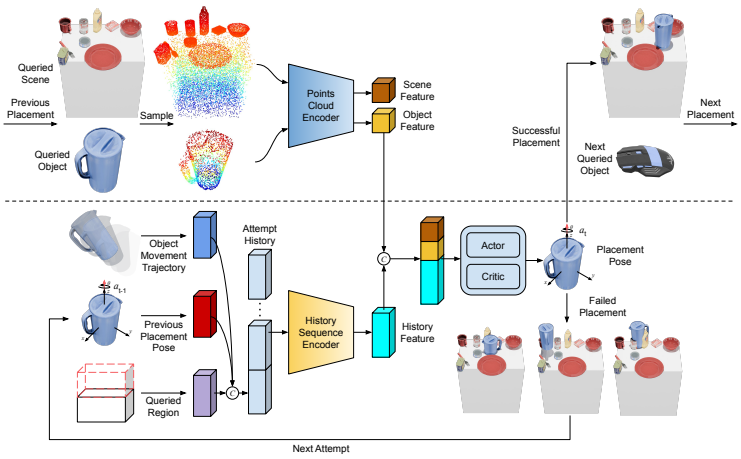

Fig. 2: **ClutterGen.** We concatenate the movement trajectory, previous placement poses, and queried regions and send them into a history sequence encoder to generate a history feature. This feature, combined with the perception feature from the point cloud encoder, is taken by ClutterGen to output the placement pose for the queried object. The simulator evaluates the placement's stability, determining whether to proceed to the next attempt or the next queried object placement.

We introduce **ClutterGen**, an auto-regressive simulation scene generator for creating physically compliant and highly diverse cluttered scenes. By framing cluttered scene generation as a reinforcement learning problem, ClutterGen learns a closed-loop policy from 3D observations without requiring pre-existing datasets or human specifications. Once trained, ClutterGen can be applied to variations of the original environment without fine-tuning. We further demonstrate the utility of ClutterGen in several downstream tasks including real-world clutter rearrangement and training robust placement policies for zero-shot sim-to-real transfer.

2 The ClutterGen Framework

Designing a simulation environment for robot training typically involves a human expert observing the scene and object geometry, deciding on a placement, running simulations to check for collision and unstable pose issues, and adjusting placements as needed. This iterative process, repeated until object poses are finalized, is difficult to scale due to its heavy human involvement and time-consuming trial-and-error. We propose **ClutterGen** to automate the steps using a single learning agent. Fig. 2 shows an overview of our method.

Observation and Action Space ClutterGen takes in the point cloud of the queried scene and object. If the current object placement fails, we inform the policy about its past actions and their impacts by providing a history of the queried object's movement trajectories. Each movement includes the object's position, orientation quaternion, and linear/angular velocity at each step. The concatenation of geometry and history embedding is the final input to our RL agent. Our policy outputs the object placement 3D translation and z-axis rotation relative to the queried region.

Policy Design We optimize our RL agent with PPO [25] algorithm. We choose beta distribution as the policy distribution, which offers several benefits. First, our design requires bounded continuous action space, and the beta distribution is well-defined among $[0,1]$. Second, beta distribution can represent more distribution shapes than normal distribution, which is essential to improve the diversity of our scene generation. As shown in Sec. 3.1, our results demonstrate that the policy trained with the beta distribution outperforms the policy trained with the squashed normal distribution in both success rate and scene diversity.

Reward Function For each placement attempt, ClutterGen optimizes the reward function $R_i = -c\sum_{i=0}^{k}(||v_i||_2 + ||a_i||_2) + n \cdot \mathbb{1}_{\text{stable}} \cdot R_0$ to minimize the accumulated absolute values of the velocity and accelerate during new object placements. Where c is a scaler to adjust the velocity and acceleration penalty, $v_i \in \mathbb{R}^6$ and $a_i \in \mathbb{R}^6$ represents the velocity and acceleration of the queried object at i_th simulation step, the indicator function $\mathbb{1}_{\text{stable}}$ will equal to 1 if the placement pose is stable otherwise will equal to 0, n represents the current queried object is the n_{th} object for the queried scene, and R_0 is a scalar reward.

3 Experiments

In this section, we evaluate the scene generation performance and investigate the generalizability of ClutterGen. We then conduct several real-world experiments to demonstrate the effectiveness of ClutterGen for downstream robotics tasks such as clutter rearrangement and stable placement policy training.

Method	Object Group									
	Group 1		*Group 2*		*Group 3*		*Group 4*		*Group 5*	
	Success Rate ↑	Stable Steps ↓	Success Rate ↑	Stable Steps ↓	Success Rate ↑	Stable Steps ↓	Success Rate ↑	Stable Steps ↓	Success Rate ↑	Stable Steps ↓
RRS	0.005	168.9±202.1	0.00	290.3±379.1	0.00	171.3±188.2	0.02	214.2±275.8	0.005	175.6±198.2
ClutterGen-OL	0.212	139.4±168.2	0.145	206.7±247.5	0.086	170.9±179.4	0.232	164.9±182.1	0.251	135.0±121.1
ClutterGen-SM	0.359	101.8±125.1	0.414	111.3±149.5	0.364	106.6±119.3	0.517	97.62±101.8	0.602	83.47±90.26
ClutterGen-Normal	**0.925**	**73.86±57.73**	0.833	85.25±91.44	0.951	**56.37±61.13**	0.969	51.19±56.22	**0.989**	**43.86±33.40**
ClutterGen	0.912	88.50±95.46	**0.874**	**70.82±97.86**	**0.963**	59.70±55.73	**0.973**	**49.89±55.55**	0.988	45.72±31.21

Tab. 1: **Success rate comparison of cluttered scene generation.** We report the average success rate and stable steps across three random seeds of training and five object groups. Our method significantly outperforms the widely adopted RRS baseline. The long-term attempt history and closed-loop design in our framework deliver the best performance.

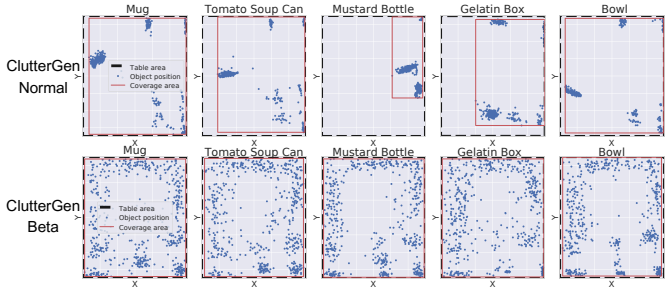

Fig. 3: **Generation diversity.** A projected view of the queried object's placements. The black dashed line represents the supporting surface area. The blue dots are queried object placement positions (x,y) across 500 setups. The red box is the coverage area, bounding all placement positions. Beta distribution greatly enhances scene diversity.

3.1 Scene Generation

Dataset We created a dataset consisting of five groups, each containing ten objects. The first four groups are selected from the PartNet-Mobility [26], Objaverse [27], and YCB [28] datasets. The fifth group, referred to as the *real group*, includes our 3D-scanned everyday objects.

Baselines 1) *Random Rejection Sampling (RRS)*: This method, which is widely adopted [16, 17, 18, 19] in recent literature, heuristically computes the position of the supporting surface in the queried scene and randomly places objects within the queried region. 2) *ClutterGen-OpenLoop (OL)*: This method uses the same architecture as ClutterGen, but without previous object movement trajectory and placement pose information for the next attempt. 3) *ClutterGen-ShortMemory (SM)*: This method uses the same architecture but only takes the latest attempt history for the next attempt. 4) *ClutterGen-Normal*: This method uses the truncated normal distribution instead of the beta distribution.

Metrics Our evaluation metrics include: 1) the *success rate* of placing all queried objects into the queried scene, and 2) the average simulation steps (*stable steps*) required for each object to achieve stability. We also access *setup diversity* by using a diversity map to evaluate the variety of the generated scenes.

Evaluation Tab. 1 shows the testing restuls with 1,000 trials. The RRS almost always failed to produce a valid object layout. This is because RRS cannot learn from active interaction with the environment. While ClutterGen-OL and CluuterGen-SM can produce some reasonable layouts, the success rate is generated low with longer simulation steps for the scene to become stable, suggesting long-term close-loop history is important. Although both ClutterGen and ClutterGen-Normal achieved high success rates in generating cluttered scenes, we assess their ability to create diverse layouts by visualizing their successful placements through a 2D projection. Each subplot in Fig. 3 shows the placement distribution for one object across 500 scenes. ClutterGen-Normal tends to place objects in very similar positions resulting in homogeneous outcomes, while our ClutterGen with beta distribution generates significantly more diverse setups.

Generalization We are interested in evaluating ClutterGen's generalization capability to generate cluttered scenes when the queried region varies during test time, even if it was fixed during training. We propose five test-time changes: 2D translation, z-axis rotation, shrink or expand xy half-extents, and random combinations of the previous changes. Our evaluation across 500 episodes with the real object group for each change is shown in Tab. 2. Overall, ClutterGen shows strong zero-shot generalization ability across all changes. To demonstrate the real-world use cases, we selected 10 furniture tables from our dataset and directly applied ClutterGen to them with all five object groups. By evaluating 50 episodes for each table, we achieved an overall 70% success rate. Qualitative examples are shown in Fig. 1. ClutterGen can naturally generate complex object relationships such as *a mug on a book* or *a fork under a plate*.

Method	*Original*	*Translation* x:$[-15cm,15cm]$ y:$[-15cm,15cm]$	*Rotation* r_z:$[-\pi,\pi]$	*Shrinkage* Δh_x:$[-10cm,0cm)$ Δh_y:$[-10cm,0cm)$	*Expansion* Δh_x:$(0cm,10cm]$ Δh_y:$(0cm,10cm]$	*Randomly Combined*
ClutterGen	0.99	0.89	0.92	0.76	0.93	0.85

Tab. 2: **Scene-level generalization results.** We report the average success rate of cluttered scene generation under various test-time changes to the queried region. ClutterGen demonstrates strong generalizability.

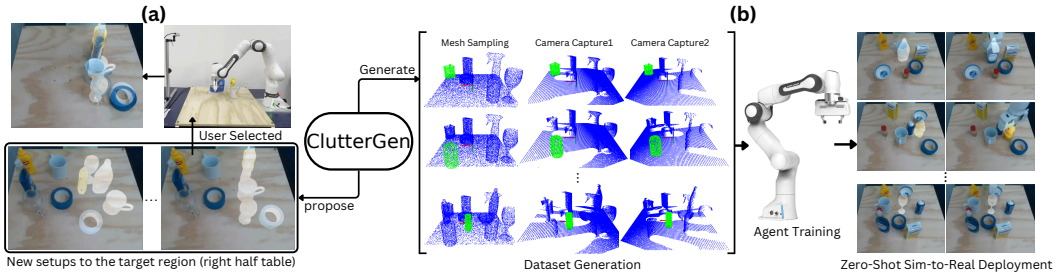

Fig. 4: **(a) Clutter rearrangement.** ClutterGen proposes several different setups for rearrangement. After a user selects a preferred setup, the Panda arm will rearrange the table accordingly. **(b) Stable Object Placement.** We generate a synthetic dataset by replaying the scene generation trajectory created by ClutterGen. This synthetic dataset is then used to train a stable object placement policy, which is directly deployed on a real robot.

3.2 Real Robotics Tasks

Clutter Rearrangement Given a clutter of objects, humans can easily determine the goal state of stable poses for each object when tasked with moving them to another location based on their preferred arrangement (e.g., from one table to another). However, enabling robots to exhibit this behavior with specific user's preferences remains challenging. For instance, when the target area is cluttered, robots must identify a safe and stable pose for new object placements to avoid collisions or simply dropping objects into the area. We demonstrate ClutterGen's applicability in this challenging task using a Franka Panda arm equipped with an RGB-D camera. Fig. 4(a) provides an overview of our approach.

We asked ClutterGen to generate ten possible layouts for the other side of the table, allowing users to select their preferred setup and obtain the target poses for each object. The robot arm then planned its motions to move the entire clutter to the target area using MoveIt [29]. Success is counted if all objects are rearranged to the user-selected setup without any collisions or unstable poses. The overall success rate was $7/10$. Failures were due to arm-object or object-object collisions during motion planning (2/3) or failed object layout proposals from ClutterGen under the ten-trial limit (1/3).

Stable Object Placement In this experiment, we leverage ClutterGen as a synthetic data generator to train a robot policy for stable object placement. Fig. 4(b) provides an overview of our approach. Our stable placement policy takes the point clouds of the queried scene and object and learns to output the stable placement pose. Specifically, we replayed the scene generation trajectory created by ClutterGen and used a virtual camera to capture the scene point cloud from different angles. We directly deployed the trained stable object placement policy to a Franka Panda arm equipped with an RGB-D camera. In each episode, the policy prediced a stable placement pose for the queried object based on the scene and the queried object points cloud. The robot arm then used such pose to plan and execute the robot's actions. Across all 50 episodes using 5 target objects, our zero-shot sim-to-real policy achieved a 72% success rate. The performance drop is likely due to significant noise in the real-world point cloud. This could be mitigated by further randomizing the dataset to simulate real-world noises or using better hardware to capture the point clouds.

4 Conclusion

In this work, we propose ClutterGen, an auto-regressive simulation scene generator for robot learning. ClutterGen efficiently generates diverse, cluttered, and physically compliant environments without relying on pre-existing datasets or human specifications. Through both simulation and real robot experiments, we demonstrate that ClutterGen can help tackle several challenging robotics tasks, such as clutter rearrangement and stable object placement in cluttered environments. Future work could be training ClutterGen with a great number of objects in the pool to enhance object-level generalization during testing.

Acknowledgments

This work is supported by ARL STRONG program under awards W911NF2320182 and W911NF2220113, by DARPA FoundSci program under award HR00112490372, and DARPA TIAMAT program under award HR00112490419.

References

[1] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[2] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[3] F. Muratore, F. Ramos, G. Turk, W. Yu, M. Gienger, and J. Peters. Robot learning from randomized simulations: A review. *Frontiers in Robotics and AI*, 9:799893, 2022.

[4] P. Katara, Z. Xian, and K. Fragkiadaki. Gen2sim: Scaling up robot learning in simulation with generative models. *arXiv preprint arXiv:2310.18308*, 2023.

[5] J. Xu, S. Song, and M. Ciocarlie. Tandem: Learning joint exploration and decision making with tactile sensors. *IEEE Robotics and Automation Letters*, 7(4):10391–10398, 2022.

[6] D. Son and B. Kim. Local object crop collision network for efficient simulation of non-convex objects in gpu-based simulators. *arXiv preprint arXiv:2304.09439*, 2023.

[7] J. Xu, H. Lin, S. Song, and M. Ciocarlie. Tandem3d: Active tactile exploration for 3d object recognition. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10401–10407. IEEE, 2023.

[8] M. Schwarz and S. Behnke. Stillleben: Realistic scene synthesis for deep learning in robotics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10502–10508. IEEE, 2020.

[9] T. Jianu, D. F. Gomes, and S. Luo. Reducing tactile sim2real domain gaps via deep texture generation networks. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8305–8311. IEEE, 2022.

[10] A. Church, J. Lloyd, N. F. Lepora, et al. Tactile sim-to-real policy transfer via real-to-sim image translation. In *Conference on Robot Learning*, pages 1645–1654. PMLR, 2022.

[11] R. Burgert, J. Shang, X. Li, and M. Ryoo. Neural neural textures make sim2real consistent. *arXiv preprint arXiv:2206.13500*, 2022.

[12] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[13] Z. Chen, A. Walsman, M. Memmel, K. Mo, A. Fang, K. Vemuri, A. Wu, D. Fox, and A. Gupta. Urdformer: A pipeline for constructing articulated simulation environments from real-world images. *arXiv preprint arXiv:2405.11656*, 2024.

[14] Y. Chen, T. Wu, S. Wang, X. Feng, J. Jiang, Z. Lu, S. McAleer, H. Dong, S.-C. Zhu, and Y. Yang. Towards human-level bimanual dexterous manipulation with reinforcement learning. *Advances in Neural Information Processing Systems*, 35:5150–5163, 2022.

[15] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024.

[16] A. Murali, A. Mousavian, C. Eppner, A. Fishman, and D. Fox. Cabinet: Scaling neural collision detection for object rearrangement with procedural scene generation. In *International Conference on Robotics and Automation*, 2023.

[17] B. Shen, F. Xia, C. Li, R. Martín-Martín, L. Fan, G. Wang, C. Pérez-D'Arpino, S. Buch, S. Srivastava, L. Tchapmi, et al. igibson 1.0: a simulation environment for interactive tasks in large realistic scenes. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7520–7527. IEEE, 2021.

[18] W. Yuan, A. Murali, A. Mousavian, and D. Fox. M2t2: Multi-task masked transformer for object-centric pick and place. In *Conference on Robot Learning*, pages 3619–3630. PMLR, 2023.

[19] A. Fishman, A. Murali, C. Eppner, B. Peele, B. Boots, and D. Fox. Motion policy networks. In *Conference on Robot Learning*, pages 967–977. PMLR, 2023.

[20] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, M. Lingelbach, J. Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pages 80–93. PMLR, 2023.

[21] C. Li, F. Xia, R. Martín-Martín, M. Lingelbach, S. Srivastava, B. Shen, K. Vainio, C. Gokmen, G. Dharan, T. Jain, et al. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *arXiv preprint arXiv:2108.03272*, 2021.

[22] Y. Wang, Z. Xian, F. Chen, T.-H. Wang, Y. Wang, K. Fragkiadaki, Z. Erickson, D. Held, and C. Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023.

[23] M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, K. Ehsani, J. Salvador, W. Han, E. Kolve, A. Kembhavi, and R. Mottaghi. Procthor: Large-scale embodied ai using procedural generation. *Advances in Neural Information Processing Systems*, 35:5982–5994, 2022.

[24] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in neural information processing systems*, 34:251–266, 2021.

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

[26] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 909–918, 2019.

[27] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023.

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

[29] D. Coleman, I. Sucan, S. Chitta, and N. Correll. Reducing the barrier to entry of complex robotic software: a moveit! case study. *arXiv preprint arXiv:1404.3785*, 2014.