RoboGen: Autonomous Skill Acquisition in Robot Simulation Based on General Language Model and Reinforcement Learning

Wang Junyang(12111028), Ji Yibing(12110501), Zhou Jingdong(12111026), Deng Haowen(12110510), Ng Wooi Cheng (12111128), Daniel Tan Sioa Hen (12111127), Mao Xinke(11910412)

***Abstract*—We present RoboGen, a generative robotic agent that automatically learns diverse robotic skills at scale via generative simulation. RoboGen leverages the latest advancements in foundation and generative models. Instead of directly using or adapting these models to produce policies or low-level actions, we advocate for a generative scheme, which uses these models to automatically generate diversified tasks, scenes, and training supervisions, thereby scaling up robotic skill learning with minimal human supervision. Our approach equips a robotic agent with a self-guided propose-generate-learn cycle: the agent first proposes interesting tasks and skills to develop, and then generates corresponding simulation environments by populating pertinent objects and assets with proper spatial configurations. Afterwards, the agent decomposes the proposed high-level task into sub-tasks, selects the optimal learning approach (reinforcement learning, motion planning, or trajectory optimization), generates required training supervision, and then learns policies to acquire the proposed skill. Our work attempts to extract the extensive and versatile knowledge embedded in large-scale models and transfer them to the field of robotics. Our fully generative pipeline can be queried repeatedly, producing an endless stream of skill demonstrations associated with diverse tasks and environments.**

I. Introduction

In recent years, robotics research has been driven by the ambitious goal of endowing robots with a diverse array of skills, enabling them to operate effectively in various non-factory settings and perform a wide range of tasks for humans. Significant advancements have been made in teaching robots complex skills such as deformable object manipulation, fluid handling, dynamic object tossing [18], in-hand re-orientation[3], and even executing high-dexterity activities like soccer playing[6] and robot parkour[19]. Despite these achievements, the acquired skills often remain isolated, with limited application horizons, and still require human-designed task descriptions and supervision for training. Additionally, the high costs and labor-intensive nature of real-world data collection have led to a reliance on simulation environments, which, although advantageous in terms of accessibility to low-level states and unlimited exploration opportunities, still demand significant effort in environment construction and task design.

This paper introduces RoboGen, a generative robotic agent designed to overcome these challenges by leveraging the latest advancements in foundation and generative models. Instead of using these models to directly produce low-level actions, RoboGen adopts a generative simulation scheme. This approach involves generating diverse tasks, scenes, and training supervisions autonomously, thus significantly scaling up robotic skill learning with minimal human intervention. The proposed system operates through a self-guided propose-generate-learn cycle, where the robotic agent first proposes interesting tasks, generates corresponding simulation environments with relevant objects and spatial configurations, and then decomposes the tasks into sub-tasks, selects optimal learning approaches, generates required training supervision, and finally learns the necessary skills. This method taps into the extensive and versatile knowledge embedded in large-scale models, transferring it effectively to the field of robotics. By employing a fully generative pipeline, RoboGen can produce an endless stream of skill demonstrations across a wide variety of tasks and environments, pushing the boundaries of automated large-scale robotic skill training.

II. Related Work

A. *Simulated Robotic Learning*

Obtaining the data necessary for modern (deep) learning algorithms directly from a real robot [9] , proves to be prohibitively expensive in terms of both time and resources, making it impractical for large-scale deployment. Hence, numerous physics-based simulation platforms have been developed previously to expedite robotics research. In the realm of robotics, physics-based simulation serves as a fast and secure method for developing, validating, and testing control algorithms and prototype designs [10]. Simulation platforms have been extensively used in the robotics community to develop a variety of skills, including object manipulation, learning robotic grasping in simulation environments [12], deep-learning-based autonomous navigation [14] , deep reinforcement learning for visual navigation, robot locomotion, learning long-horizon visual manipulation tasks in simulation [17], in-hand re-orientation, object tossing, and interaction with dynamic environments [11] .

B. *OBJAVERSE*

OBJAVERSE is a significant tool in the realm of object model generation for simulated environments, offering a wealth of high-quality, annotated 3D models that can be utilized across a wide range of applications. By sourcing objects from Sketchfab, a popular 3D marketplace, OBJAVERSE provides a diverse collection of models, each enriched with detailed metadata. This metadata includes object names, fixed category assignments, unrestricted tags, and natural language descriptions, which together facilitate the creation of realistic and contextually relevant simulations. The diversity of OBJAVERSE's dataset make it particularly valuable for several key areas, for instances, 3D object retrieval and classification, scene understanding and reconstruction, and robotic manipulation and interaction. Besides that, Objaverse significantly contributes to VR and AR applications by providing realistic 3D models that enhance the immersive experience (Wu et al. 2021). In addition, Objaverse is used to create detailed urban environments for testing and training autonomous vehicles. These realistic simulations allow for safe and extensive testing of navigation algorithms and systems without the risks associated with real-world testing [5] . On another hand, game developers can leverage Objaverse to generate high-quality game assets, creating more realistic and interactive gaming environments.

C. Large Language Model (LLM)

Large Language Models (LLMs) like GPT-4 have made significant strides in various domains, including natural language processing, machine learning, and robotics. These models have revolutionized how machines understand, generate, and interact with human language, enabling a wide range of applications from automated text generation to complex decision-making processes. LLMs have demonstrated exceptional capabilities in understanding and generating human language. For instance, the original GPT model and its successors, such as GPT-2 and GPT-3, have shown the ability to produce coherent and contextually relevant text based on given prompts [13][2] . These models have been employed in diverse tasks, including summarization, translation, and question-answering, showcasing their versatility and effectiveness in handling complex language tasks. The integration of LLMs into robotics and AI systems has opened new avenues for improving the performance and capabilities of these technologies. For example, LLMs have been used to enhance robotic planning and control by generating detailed and context-aware instructions for robotic tasks. GPT-4, as a more advanced iteration, has further expanded these capabilities. It has been utilized in robotic skill learning, where it helps generate detailed task descriptions and decompositions, facilitating the development of sophisticated robotic behaviors [1] . The use of GPT-4 in such contexts demonstrates its potential to bridge the gap between high-level language instructions and low-level robotic actions, making robotic systems more intuitive and adaptable. Recent works have explored the use of LLMs for low-level control actions and goal specification in robotics. These models can translate high-level language instructions into precise control commands, enabling more accurate and reliable execution of tasks [15] . Furthermore, LLMs like GPT-4 have been employed to specify and refine goals for AI systems, ensuring that they align with the desired outcomes and adapt to changing requirements [8] .

The advancements in LLMs, including GPT-4, have significantly impacted various fields, particularly in enhancing the capabilities of AI and robotic systems. By providing sophisticated language understanding, task generation, and data augmentation, these models facilitate the development of more intuitive, adaptable, and effective technologies. As LLMs continue to evolve, their integration into different domains is expected to drive further innovations and breakthroughs, making them indispensable tools in the modern technological landscape.

III. RoboGen

RoboGen is an automated pipeline that utilizes the embedded common sense and generative capabilities of the latest foundation models for automatic task, scene, and training supervision generation, leading to diverse robotic skill learning at scale. We illustrate the whole pipeline in Figure 1, composed of several integral stages: Task Proposal, Scene Generation, Training Supervision Generation, and Skill Learning. We detail each of them in the following.

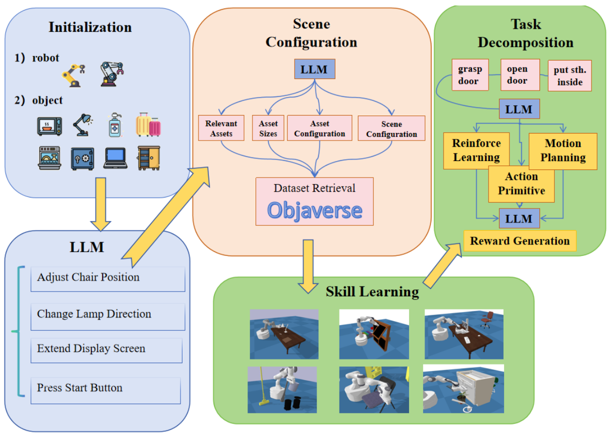

Fig. 1. RoboGen: An Integrated Framework for Initialization, LLM, Scene Configuration, Skill Learning, and Task Decommposition

A. Task Proposal

RoboGen uses GPT-4 to generate tasks involving the interaction and manipulation of objects by a robotic arm. Objects are initialized from a predefined list that includes common household items such as ovens, microwaves, and laptops from the PartNetMobility[16] and RLBench[7]datasets. A query is constructed with the object's category, articulation tree, and semantic annotations, asking GPT-4 to return tasks including the task name, description, additional required objects, and relevant joints/links. For example, GPT-4 might generate the task ”heat up a bowl of soup” for a microwave, specifying the joints and links to interact with. The task description might be: “The robot arm places a bowl of soup inside the microwave, closes the door, and sets the microwave timer for an appropriate heating duration.” Additional objects needed for the generated task could include “A bowl of soup,” and task-relevant joints and links could include joint 0 (for opening the microwave door), joint 1 (for setting the timer), link 0 (the door), and link 1 (the timer knob) [15]. Note that for cases where we sample non-articulated objects or use example-based initialization, the sampled objects and examples are provided only as a hint for task proposal, and the generated tasks will not be tied to them For articulated objects, since PartNetMobility is the only articulated object dataset with high quality, and already covers diverse range of articulated assets, we will generate tasks dependent on the sampled asset. For locomotion and soft-body manipulation tasks, we use only example-based initialization, and resort to GPT-4 to populate additional required objects. By repeatedly querying with different sampled objects and examples, we can generate a diverse range of manipulation and locomotion tasks, concerning the relevant object affordances when needed[15].

B. *Simulated Scene Generation*

In the realm of computer vision and robotics, the creation of realistic and varied simulated environments is a cornerstone for advancing research and development. Simulated scenes provide a controlled setting where robots and AI systems can be trained, tested, and refined without the constraints and unpredictability of the physical world. Once a task is proposed, GPT-4 is used to generate additional queries to populate the scene with semantically relevant objects, creating complex and diverse environments. Retrieved objects can come from existing databases or be generated via text-to-image and image-to-3D model generation. In this case, the 3D-models of the objects are obtained from database (Objaverse). Additionally, GPT-4 verifies these objects’ sizes to ensure physical plausibility.

1) ***Object Model Genration*** *:* We utilize Objaverse for object model generation, leveraging its extensive repository and sophisticated annotation framework to create diverse and detailed objects . OBJAVERSE is an extensive annotated 3D dataset designed to facilitate research across various domains within computer vision [4]. The objects in this dataset are sourced from Sketchfab, an online 3D marketplace where users can upload and share models for both free and commercial purposes. The objects that match the objects models in Objaverse will be downloaded and hence, generated in the simulated environment.

When objects are uploaded to Sketchfab and subsequently included in OBJAVERSE, they come with a comprehensive set of foundational annotations provided by their creator. This metadata encompasses several key elements. Firstly, it includes the object’s name, which serves as a primary identifier. Secondly, each object is assigned to a set of fixed categories, facilitating structured organization and easier discovery within the platform. Additionally, creators can attach a diverse array of unrestricted tags, which enhance the searchability and contextual relevance of the objects. Finally, each object is accompanied by a natural language description that provides detailed insights and background information, enriching the user’s understanding and engagement with the object.

For instance, when a task is assigned to GPT-4 as ”Move the chair from the initial position to another place”, the keyword in the task description such as ”chair”, which identifies the object within the simulation scene, will be selected and matched to the relevant categories in OBJAVERSE, and spawn the object model in the simulation environment. In the example, a robot arm will also be modelled since the keyword ”move” appears in the sentence, indicating the need for an agent to perform the action.

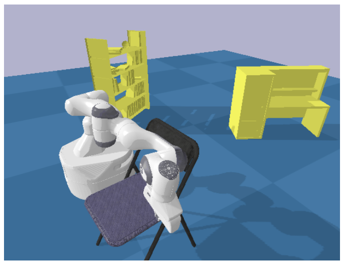

Fig. 2. Task Description: ”Move the chair to the desired position”. Two main objects, the chair and the robot arm, are generated.

2) ***Relative Size Configuration*** *:* Relative size configuration is crucial for accurately modeling tasks in simulation environments. For instance, if the task description is ”put a pen into a box,” the pen must be appropriately sized to fit inside the box, implying it should be smaller than the box. This relative size determination ensures the feasibility of the task within the simulated context. GPT-4 can handle this configuration by analyzing the task description and adjusting the dimensions of the objects accordingly. By interpreting the relationship between objects, GPT-4 can ensure that the pen is modeled to be smaller than the box, facilitating accurate and realistic task execution.

3) ***Initial Scene Configuration*** *:* Initial scene configuration can be performed by utilising GPT-4, setting articulated objects in valid states, allowing the robot to effectively learn and execute specific tasks. As an example, if the task is to ”open a door,” as shown in Fig. 3., the door should be initialized in the closed position to provide a clear action target for the robot. Similarly, for the task of ”closing a drawer”, the drawer must

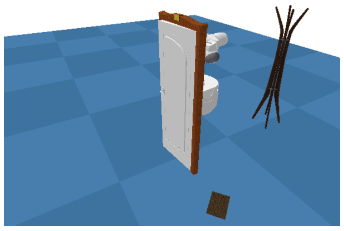

Fig. 3. Task Description: ”Opening a door”. The door is initially closed since the task for the robot arm is to open the door.

be initially open. Additionally, for the task of ”putting the pen on the table," the pen should be initialized in a location away from the table. These starting states ensure that the robot can perform the required actions accurately. By utilizing GPT-4, we can configure these initial states, specifying the joint angles of articulated objects to meet the necessary conditions. This careful setup is essential for the robot to learn and perform the desired manipulations effectively.

C. Reinforcement Learning using Ray RLlib

This section outlines the methodology employed to train and evaluate RL agents for robotic manipulation using the Ray RLlib library.Ray's scalability and flexibility make it an ideal framework for this purpose, allowing efficient parallel training and robust experiment management.We focus on Soft Actor-Critic (SAC) RL algorithms, and leverage Ray's features to enhance the training pipeline.

Our RL training pipeline consists of several key components designed to facilitate efficient training and evaluation of robotic manipulation agents using Ray.These components include custom logging, environment configuration, policy management, training routines, and performance evaluation.Ray's capabilities in distributed computing and experiment management play a crucial role in optimizing these components.

*1) **Custom Logging Mechanism**:* To ensure organized and efficient tracking of different training sessions, we implemented a custom logging mechanism using Ray's UnifiedLogger.This system generates unique log directories based on the current timestamp and a custom identifier string, facilitating easy retrieval and analysis of training progress.Ray's logging infrastructure integrates seamlessly with our custom logger, providing a robust solution for managing logs.

*2) **Configuration Setup for RL Algorithms**:* We developed a flexible configuration setup function to tailor the RL algorithms to the specific requirements of the robotic manipulation tasks.This function adjusts hyperparameters such as batch size, the number of workers, and the architecture of the neural networks used by the RL algorithms.Specifically, we configure PPO and SAC with parameters optimized for our tasks, ensuring robust and efficient learning. Ray's configuration management capabilities allow us to easily adapt and experiment with different settings.

*3) **Policy Loading and Management**:* Policy loading and management are critical for initializing RL agents and resuming training from previously saved states. Our methodology includes functions to load existing policies from checkpoints and restore them accurately using Ray's checkpointing features. This capability allows for continuous training and fine-tuning of policies, which is essential for improving agent performance over time.

*4) **Training Routine with Periodic Evaluations**:* The training routine forms the core of our RL pipeline. During training, the agent interacts with the environment to collect experiences and updates its policy accordingly. Ray's ability to scale across multiple CPUs and GPUs enables efficient parallel training, significantly speeding up the learning process. To monitor progress and prevent overfitting, we conduct periodic evaluations at predefined intervals. During these evaluations, we assess the policy's performance, save the best-performing models, and log key metrics such as total timesteps, mean rewards, and training duration. The training process also includes mechanisms to manage and store the best models based on evaluation performance.

*5) **Visualization of Agent Performance**:* To gain insights into the agent's behavior and performance, we implemented a policy rendering function. This function visualizes the agent's actions within the environment, allowing us to observe how the trained policy operates in real-time. Visualization is a crucial step for qualitative assessment and debugging, providing a clear understanding of the agent's capabilities and areas for improvement. Ray's support for custom environments and visualization tools enhances our ability to render and analyze agent behavior effectively.

*6) **Environment Configuration and Initialization**:* A critical aspect of our methodology is the setup and initialization of the simulation environment. We developed a function to configure the environment based on specific task parameters, ensuring that it is correctly instantiated with the necessary settings. This function handles various aspects of the environment, including task configurations, action spaces, and rendering options. Proper environment setup is essential for accurate training and evaluation of the RL agent. Ray's environment registration and configuration capabilities simplify this process, allowing seamless integration with our training pipeline.

Our methodology has been applied to train RL agents for various robotic manipulation tasks. Ray's scalability and distributed computing capabilities have been instrumental in efficiently handling large-scale training sessions. The custom logging mechanism and periodic evaluations have been essential in tracking progress and ensuring the reliability of the training process. The flexible configuration setup has allowed us to optimize the algorithms for different tasks, leading to improved performance and efficiency. The visualization of agent performance has provided valuable insights into the agent's decision-making process, helping us identify and address potential issues. Overall, the combination of these components, powered by Ray's robust infrastructure, has enabled us to develop robust and effective RL agents capable of performing complex manipulation tasks in simulated environments.

IV. EXPERIMENTS

RoboGen is an automated pipeline that can be queried endlessly, and generate a continuous stream of skill demonstrations for diverse tasks. In our experiments, we aim to answer the following questions: leftmargin=*,label=

- **Task Diversity:** How diverse are the tasks proposed by RoboGen robotic skill learning?
- **Scene Validity:** Does RoboGen generate valid simulation environments that match the proposed task descriptions?
- **Skill Learning:** Can RoboGen learn the generated task?

A. EVALUATION METRICS AND BASELINES

We use the following metrics and baselines for evaluating our system:

1) Task Diversity: The diversity of tasks generated by task diversity can be measured from multiple aspects, such as the semantics of the task, the scene configuration of the generated simulation environment, the appearance and geometric shape of the retrieved object assets, and the robot actions required to execute the task. For the semantics of tasks, we quantitatively evaluate them by calculating the self BLEU and similarity of the generated task descriptions, with lower scores indicating better diversity. We compared with established benchmarks, including RLBench (James et al., 2020), Maniskill 2 (Gu et al., 2023), Meta World (Yu et al., 2020), and Behavior-100 (Srivastava et al., 2022). For object assets and robot actions, we use generated simulation environments and visualizations of learned robot skills to qualitatively evaluate RoboGen.

2) Scene Validity (CLIP): Calculate normalized Frame-Text Similarity Score (FTSS) for tasks with object verification using the CLIP model. CLIP (Contrastive Language-Image Pretraining) aligns images and text by encoding them into a shared embedding space, allowing us to measure the similarity between GIF frames and textual descriptions. The FTSS is calculated by encoding both the frames and the text, computing cosine similarity between them, and normalizing the scores to enable meaningful comparisons across different datasets. This process highlights the effectiveness of our object verification in maintaining scene validity, as our scores demonstrate strong alignment between visual and textual content when compared to other projects.

3) Skill Learning: To evaluate skill learning performance, we conducted a comparative analysis by directly observing the reward functions generated by RoboGen for various tasks. RoboGen simultaneously employs motion planning-based primitives, gradient-based trajectory optimization, and reinforcement learning to acquire skills. For each task, we executed the combined method using four different random seeds and reported the mean and standard deviation of the task returns. The reward functions for the evaluated tasks were manually verified to ensure correctness.

B. RESULTS

1) Task Diversity: The quantitative evaluation results are shown in Table 1. We compared the RoboGen versions that generated a total of 14 tasks. As shown in the figure, RoboGen achieved the lowest SelfBLEU and embedding similarity compared to all previous benchmarks, indicating that under these two indicators, the diversity of generated tasks is higher than the manually established benchmarks previously. This indicates that RoboGen can generate a set of tasks whose diversity matches or exceeds previously handcrafted skill learning benchmarks and datasets.

2) Scene Validity (CLIP): Figure 3 illustrates the normalized Frame-Text Similarity Scores (FTSS) for RoboGen, the ViT-L-14 ensemble, and the CLIP-S score on the Flickr8K

	RoboGen	Behavior-100	RLbench	MetaWorld	Maniskill2
Number of tasks	30	50	53	50	20
Self-BLEU↓	0.134	0.299	0.317	0.322	0.674
Embedding Similarity↓	0.061	0.210	0.200	0.263	0.194

dataset. The quantitative evaluation demonstrates that RoboGen achieves an average FTSS of 0.694, outperforming the ViT-L-14 ensemble (0.688) and significantly higher than the CLIP-S score on the Flickr8K dataset (0.512). These results highlight the effectiveness of the object verification process implemented in RoboGen, ensuring a strong alignment between GIF frames and textual descriptions. The high FTSS for RoboGen indicates superior performance in maintaining scene validity compared to the other benchmarks. The consistency and reliability of RoboGen's task generation underscore its capability to produce diverse and contextually relevant scenes, surpassing the previously established benchmarks. This comprehensive analysis confirms that RoboGen can generate tasks with high validity, aligning closely with textual descriptions and setting a new standard for scene generation in automated systems. Future work will involve further comparative studies, including tasks without verification steps, to validate the robustness and versatility of RoboGen's approach.

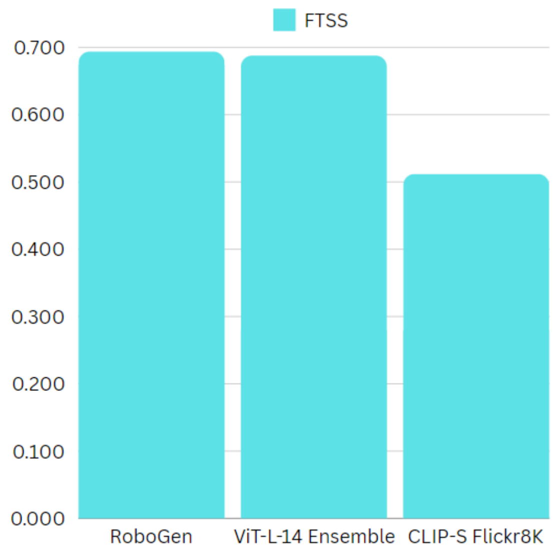

Fig. 4. Comparison of Normalized FTSS

3) Skill Learning: We evaluate the performance on multiple tasks involving different skill requirements. The findings indicate that incorporating object and size verification significantly enhances performance across tasks. When these verification methods are ablated, the scores drop notably, demonstrating their critical role in task completion. To evaluate learning performance, we generated several reward function images.

Most tasks exhibit typical reinforcement learning trends, where the reward is initially low, gradually increases, and then stabilizes as training progresses. For instance, tasks like "close the window" and "open the laptop screen" show curves that rise sharply from negative values and then level off.

Each task's reward curve includes both an average reward (solid line) and a reward range (shaded area). In the early stages of training, the reward range is generally wide, indicating high variability in rewards. As training continues, the reward range narrows, suggesting that rewards become more consistent.

Some tasks exhibit highly irregular reward curves, indicating poor learning performance. For example, tasks like "press the start button" and "turn the hot water switch" show erratic and unstable reward patterns throughout the training process. These anomalies suggest that the agent struggles to learn effective strategies for these tasks, leading to inconsistent and suboptimal rewards.

The reward function graphs indicate that task complexity and environmental variations significantly affect the agent's learning performance and reward convergence speed. We also experimented with using only reinforcement learning (RL) without RoboGen, and the final learning performance was significantly lower than that achieved with RoboGen. This demonstrates that RoboGen, which integrates motion planning-based primitives, gradient-based trajectory optimization, and reinforcement learning, provides a much better overall learning outcome. Future research can explore strategies to optimize training for these tasks to accelerate the learning process and improve reward stability.

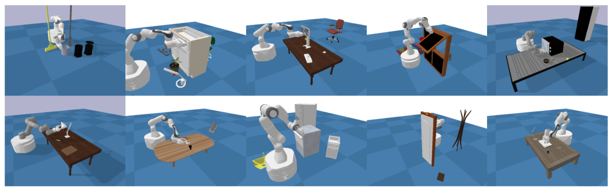

Fig. 5. 10 examples of GIF output

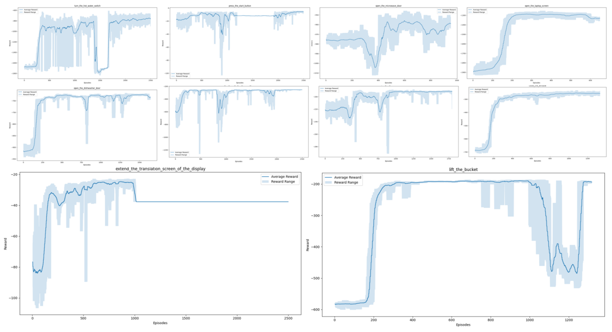

Fig. 6. 10 examples of reward function

Based on the reward function images and GIF images of these ten examples, we can manually judge that eight of the tasks can be learned and completed successfully. Among the tasks that were learned, most showed a significant change in the reward function around 200 episodes and eventually stabilized within ±5% of this value. For the tasks that were not learned, we believe this is due to the higher complexity of the actions required. Overall, the learning success rate is above 80%.

V. Conclusion

This paper introduces RoboGen, a generative agent designed to autonomously propose and learn diverse robotic skills through generative simulation. RoboGen leverages the latest advancements in foundational models to automatically generate diverse tasks, scenes, and training supervision, thereby scaling up robotic skill learning with minimal human intervention once deployed. This fully generative pipeline can be continuously queried, producing a large volume of skill demonstrations across various tasks and environments. The system is agnostic to the backend foundational models, allowing for continuous upgrades as more advanced models become available.

Despite RoboGen's promising capabilities, the current system still has several limitations. First, large-scale validation of learned skills, i.e., verifying if the resultant skill truly accomplishes the corresponding task based on text descriptions, remains a challenge. Future improvements in multimodal foundational models could address this issue. Second, there is an inherent sim-to-real gap when deploying the learned skills in real-world scenarios. However, ongoing advancements in physically accurate simulation, domain randomization, and realistic sensory signal rendering are expected to narrow this gap. Third, the system assumes that existing policy learning algorithms are sufficient for learning the proposed skills given the right reward functions. However, it has been observed that the robustness of these algorithms is still limited, often requiring multiple runs to produce successful skill demonstrations for certain tasks. Future work will focus on integrating more powerful policy learning algorithms with better action parameterizations into RoboGen.

Acknowledgments

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[3] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In *Conference on Robot Learning*, pages 297–307. PMLR, 2022. URL http://proceedings.mlr.press/v100/chen20a.html.

[4] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects, 2022.

[5] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator, 2017.

[6] Tuomas Haarnoja, Ben Moran, Guy Lever, Sandy H Huang, Dhruva Tirumala, Markus Wulfmeier, Jan Humplik, Saran Tunyasuvunakool, Noah Y Siegel, and et al. Hafner, Roland. Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *arXiv preprint arXiv:2304.13653*, 2023. URL https://arxiv.org/abs/2304.13653.

[7] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark and learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

[8] Ivan Kapelyukh, Yifei Ren, Ignacio Alzugaray, and Edward Johns. Dream2real: Zero-shot 3d object rearrangement with vision-language models. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*, 2023.

[9] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection, 2016.

[10] C. Karen Liu and Dan Negrut. The role of physics-based simulators in robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1). doi: 10.1146/annurev-control-072220-093055. URL https://par.nsf.gov/biblio/10276894.

[11] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation, 2019.

[12] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours, 2015.

[13] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9, 2019.

[14] Lei Tai, Shaohua Li, and Ming Liu. A deep-network solution towards model-less obstacle avoidance. pages 2759–2764, 10 2016. doi: 10.1109/IROS.2016.7759428.

[15] Yen-Jen Wang, Bike Zhang, Jianyu Chen, and Koushil Sreenath. Prompt a robot to walk with large language models. *arXiv preprint arXiv:2309.09969*, 2023.

[16] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. Sapien: A simulated part-based interactive environment, 2020.

[17] Adriel Yeo, Benjamin Kwok, Angelene Joshna, Kan Chen, and Jeannie Lee. Entering the next dimension: A review of 3d user interfaces for virtual reality. *Electronics*, 13:600, 02 2024. doi: 10.3390/electronics13030600.

[18] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4):1307–1319, 2020. doi: 10.1109/TRO.2020.2972567. URL https://ieeexplore.ieee.org/document/8968424.

[19] Ziwen Zhuang, Zipeng Fu, Jianren Wang, Christopher Atkeson, Soeren Schwertfeger, Chelsea Finn, and Hang Zhao. Robot parkour learning. *arXiv preprint arXiv:2309.05665*, 2023. URL https://arxiv.org/abs/2309.05665.