RCareGen: An Interface for Scene and Task Generation in RCareWorld

Shuaixing Chen
Shanghai Jiao Tong University
Shanghai, China
alkdischen@sjtu.edu.cn

Ruolin Ye
Cornell University
Ithaca, NY, United States
ry273@cornell.edu

Saurabh Dingwani
Arizona State University
Tempe, AZ, United States
sdingwan@asu.edu

Pooyan Fazli
Arizona State University
Tempe, AZ, United States
pooyan@asu.edu

Hasti Seifi
Arizona State University
Tempe, AZ, United States
hasti.seifi@asu.edu

Tapomayukh Bhattacharjee
Cornell University
Ithaca, NY, United States
tapomayukh@cornell.edu

***Abstract*—This late-breaking report presents RCareGen, a graphical interface that integrates natural language commands with RCareWorld, a physics simulator for robotic caregiving scenarios. RCareGen has three core modules: (1) a front-end web interface, (2) an LLM-based code generator, and (3) the RCareWorld simulation backend. The front-end web interface enables novice users to input natural language, which is translated into Python code by an LLM-based code generator. This generated code interacts with RCareWorld APIs to run the simulation backend, facilitating scene setup, modifications, simple movements, and human-robot interaction tasks. Additionally, the system supports iterative feedback, allowing users to refine scenes and tasks interactively. By simplifying simulation setup and enhancing task diversity, RCareGen introduces a novel interface that democratizes robot simulation and programming across diverse domains.**

***Index Terms*—Large Language Models, Robotics Simulator, Scene Generation**

I. Introduction

Recent advances in simulation platforms have significantly improved human-robot interaction research [1–3]. However, significant challenges often hinder the adoption of such platforms. First, these platforms have a steep learning curve, requiring users to master specialized tools such as Unity or Python, which can be overwhelming for non-experts. Second, setting up tasks and environments is labor-intensive and often results in limited task diversity and creativity.

Our key insight is that large language models (LLMs) can lower the barrier to using simulation platforms by simplifying interactions, allowing users to set up tasks and explore scenarios effortlessly through intuitive natural language commands. Leveraging this insight, we developed RCareGen, a system that integrates ChatGPT-3.5 [4] with RCareWorld [1], which is a state-of-the-art simulation platform for caregiving robots, to simplify simulation environment setup and task execution. RCareGen consists of three core components: (1) a front-end web interface for natural language interactions, (2) an LLM-based code generator that translates prompts into Python code, and (3) the RCareWorld simulation backend, which executes the generated code to create and run the simulation environment.

Users' interaction with RCareGen begins with *scene generation*. They can describe their desired scene using high-level prompts such as, "Set up a scene with a Franka robot, a table, and a cup." in the web interface. The LLM-based code generator processes this input to Python APIs that retrieve and load relevant assets into the simulation. Users can refine the scene iteratively by providing feedback, such as, "Move the cup 10 cm to the left." With the generated scene, users can then perform *task generation*. By providing an instruction such as "Train a PPO algorithm to let the robot move the cup forward by 10cm", the LLM-based code generator will generate the code to train and execute the skill for the task.

Through various tasks, we demonstrate how RCareGen bridges the gap between natural language inputs and executable logic, enabling non-experts to configure and engage with complex simulation environments. RCareGen showcases the potential of LLMs to democratize access to robotics simulation platforms and empower users across a wide range of domains.

We summarize our contributions as the following:

- A platform-agnostic web interface interacting with novice users using natural language.
- An LLM-based code generator that translates natural language to Python code.
- RCareGen, an interface enabling natural language-based scene and task generation for human-robot interaction.
- Demonstrations highlighting the ability to set up scenes, modify them dynamically using user feedback, generate simple robot movements, and create human-robot interaction tasks.

II. Related Work

A. Code Generation with LLM

Recent work highlights the potential of LLMs to generate executable code for robotics. Code-As-Policies [5] demonstrates

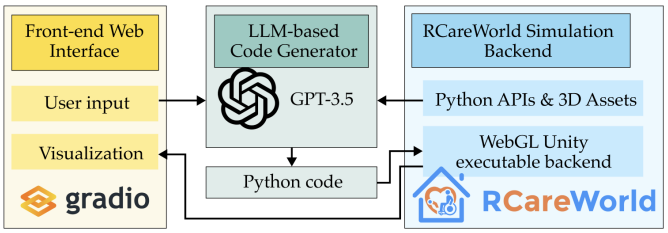

Fig. 1: **Overview of the RCareGen system architecture**, demonstrating the interaction between the front-end web interface, LLM-based code generator, and the RCareWorld simulation backend.

that LLMs can translate natural language instructions into executable robot policies. ProgPrompt [6] introduces specialized prompting techniques to enhance the reliability of code generation for robotics. RoboGen [7] focuses on generating diverse robot behaviors for manipulation tasks in simulated environments. Our RCareGen leverages the realistic human avatars in RCareWorld [1], enabling seamless generation of human-robot interaction tasks.

B. Scene Synthesis in Simulation

Scene synthesis and environment generation in robotics simulation is an active area of research. iGibson [8] introduces a framework for creating interactive environments to support robot learning. AI2-THOR [9] develops methods for procedural generation of indoor environments. Both platforms emphasize the importance of diverse, configurable environments for robotics research but often rely on extensive manual setup. Recent advancements focus on automating scene generation. 3D-SIS [10] introduces methods to automatically generate plausible indoor environments, while BEHAVIOR-1K [11] generates task-relevant environments to enhance robot learning. RCareGen provides an interface to autonomously generate the scene, as well as an interface to incorporate user feedback, making the scene generation more controllable.

III. Generating Simulation Scene and Task using RCareGen

A. System Overview

RCareGen features three key components that enable natural language-driven scene and task generation as shown in Figure 1:

1) **Front-end Web Interface**: This interface takes in natural language instructions from users and visualizes the simulation scene, providing an intuitive entry point for interaction. Being web-based, it is platform-agnostic, ensuring accessibility for users across various devices.
2) **LLM-Based Code Generator**: Powered by ChatGPT-3.5 [4], this component translates natural language prompts into executable Python code using a prompt template (detailed in Section III-B) to ensure reliability and consistency.
3) **RCareWorld Simulation Backend**: This module executes the generated code, handling asset loading, scene configuration, and task execution.

Using the three components, the system allows users to iteratively refine scenes or task parameters with natural language commands.

B. Prompt Specification

We provide the LLM-based code generator with a structured prompt as follows. The prompt describes (a) the role of GPT, (b) the available APIs in RCareWorld (including the input to the function and the returned arguments, and available assets), (c) objects available in the scene, (d) the format of the output, (e) example of a relevant code generation task with user input, and (f) user inputs, such as the robot and objects wanted in the scene.

```
Example Prompt (a) Role of GPT You are a robot
control assistant. You should generate Python
code using only these APIs:... (b) Available APIs:
get_object_position...move_to_position... (c) Scene
description ...Box 1 position is: x=0.4, y=0.1,
z=0.1... (d) Output format The format of output should
be a Python file..... Use only the provided APIs... (e)
Examples Below is a simple example of generating code
using RCareWorld APIs (f) User input I want to set up a
scene with...
```

C. Frontend-Backend Integration

We leverage Gradio [12] to implement the web interface. The interface includes a Unity WebGL container for visualization and real-time camera controls, enabling seamless interaction with loaded assets as shown in Figure 2. The RCareWorld backend is a pre-built WebGL executable file with the assets (e.g., human avatars, various robot arms, household objects, etc.). The frontend interface loads the backend simulation in the WebGL container.

D. User feedback

RCareGen implements an iterative feedback mechanism that allows users to provide feedback to modify the generated scenes through multiple rounds of conversations.

The feedback mechanism involves three key stages:

1) **Intent Understanding:** The LLM first analyzes the feedback context, considering the current scene state and previous feedback if applicable. This contextual understanding helps distinguish between absolute commands (e.g., "Move the cup 10cm to the left") and relative adjustments (e.g., "Move the cup closer to the robot"). If the goal is not clearly given, the system performs spatial reasoning in the next step.
2) **Spatial Reasoning:** The system maintains a structured representation of object relationships and constraints for spatial feedback. When receiving feedback such as "The

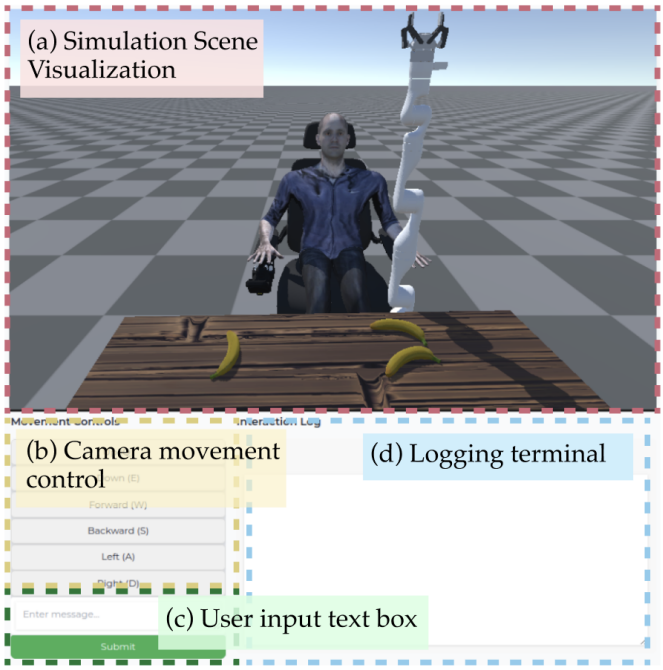

Fig. 2: The **front-end web interface** of RCareGen with (a) a Unity scene displaying the robot and objects, (b) movement control buttons for the camera to adjust the rendered viewpoint, (c) a text box to get user inputs, and (d) a logging terminal.

cup is too far from the robot", the system calculates appropriate intermediate positions.

3) **Incremental Refinement:** Instead of treating each feedback as an independent command, the system maintains a history of adjustments to ensure continuous and coherent scene evolution.

IV. Demonstration

We present three core capabilities of RCareGen: (1) scene setup and modification, (2) generating simple movements, and (3) creating human-robot interaction tasks. For each capability, we demonstrate a user prompt, the corresponding generated code, user feedback, and the refined code. We also provide a visualization of the simulation scene. For further details, please refer to the supplementary video included with the demo.

A. Demonstrating Scene Setup and Modification

This example shows that the user can configure and modify a scene dynamically using natural language instructions. The user starts by prompting with "`Set up an environment with a Kinova robot, several bananas, a mug, and a table. A care recipient sits next to the table.`" With this prompt, the LLM generates the following code:

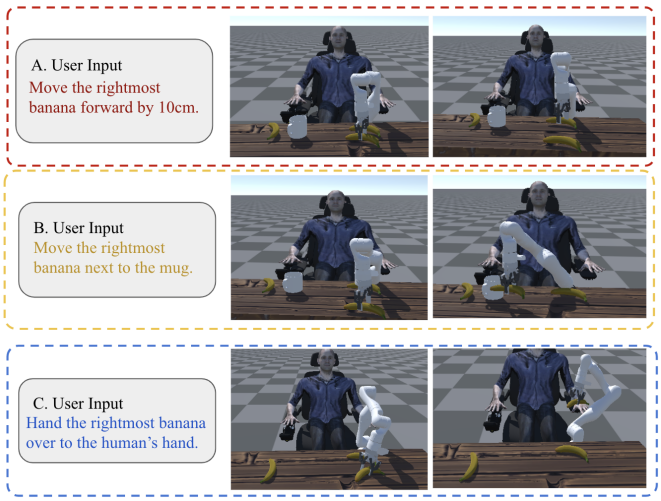

Fig. 3: Demonstration of three interaction sequences: (a) Scene modification where the user requests moving a banana forward; (b) Robot manipulation task where the robot picks and places the banana next to the mug; and (c) Human-robot interaction where the robot hands over the banana to the human avatar. Each sequence shows the user input (left) and corresponding simulation states (middle, right).

```
load_asset("kinova_robot", position=[0, 0, 0])
load_asset("banana1", position=[0.2, 0.1, 0.0])
load_asset("banana2", position=[-0.2, 0.1, 0.0])
load_asset("banana3", position=[-0.2, 0.05, 0.0])
load_asset("mug", position=[0.3, 0.1, 0.0])
load_asset("table", position=[0.0, 0.1, 0.0])
```

The generated code initializes a scene with a Kinova robot, several bananas, a mug, and a table placed at the respective positions, and a care recipient sitting by the table. The user then gives feedback, saying, "`Move the rightmost banana forward by 10 cm.`" It leads to the following generated code:

```
# Obtain the banana position
pos = get_object_position("banana1")
# Grasp the banana
pick_object("banana1")
# Move the banana forward by 10cm
move_to_position(pos[0], pos[1] + 0.1, pos[2])
```

This moves the rightmost banana forward by 10cm in the simulation, as shown in Figure 3a.

B. Demonstrating Programming Simple Movements

This example shows that the user can program robot actions using natural language instructions after configuring the scene.

The user starts by prompting "Move the rightmost banana next to the mug." LLM leverages interpolation tools in RCareWorld to plan a trajectory using the following code:

```
# Grasp the banana
pick_object("banana1")
# Obtain the mug position
mug_pos = get_object_position("mug")
# Place the banana next to the mug
place_object(mug_pos[0], mug_pos[1] + 0.05,
    mug_pos[2])
```

As shown in Figure 3b, the robot successfully picks up the banana and places it next to the mug.

C. Demonstrating Human-Robot Interaction Tasks

Leveraging the human avatars in RCareWorld, RCareGen seamlessly supports human-robot interaction tasks. With the prompts for a handover task: "Hand the rightmost banana over to the human's hand" it generates the following code:

```
# Obtain the hand position
hand_pos = get_object_position("left_hand")
# Grasp the banana
pick_object("banana1")
# Move to the hand
move_to_position(hand_pos[0], hand_pos[1] -
    0.1, hand_pos[2] + 0.1)
# Handover
gripper_control(True)
move_to_position(hand_pos[0], hand_pos[1] -
    0.3, hand_pos[2] + 0.1)
```

This code programs the robot to hand over the banana to the human hand, as shown in Figure 3c.

V. Evaluation

We evaluated the system's performance in terms of response timing by running each demonstration scenario described in Section IV ten times. The tests were conducted on a workstation equipped with a 12th Gen Intel Core i7-12700 CPU, an NVIDIA RTX 4060 GPU, and 32 GB of RAM. The time step in RCareWorld was set to 0.02 seconds. The scene setup and modification tasks showed an average response time of 2.8 seconds (±0.4 seconds), with scene loading completed in 1.2 seconds (±0.2 seconds). Simple robot movement commands had an average processing time of 3.2 seconds (±0.5 seconds), with actual execution taking 1.8 seconds (±0.3 seconds). Human-robot interaction tasks required slightly longer processing times, averaging 3.5 seconds (±0.6 seconds) for response generation and 2.1 seconds (±0.4 seconds) for execution.

We then invited five users (4 female and 1 male, aged 21–53, mean age 27) to interact with the system and provide feedback. Users rated the interface on a 10-point Likert scale across three dimensions: ease of use (8.6/10), usefulness (8.2/10), and overall satisfaction (8.4/10). They particularly appreciated the system's ability to support natural language interaction and the immediate visual feedback provided by the simulation environment. Users also found the iterative feedback mechanism helpful in refining their intentions. However, some noted that expressing more complex spatial relationships solely through natural language could be challenging.

These initial results indicate that RCareGen achieves reasonable response times for interactive use while delivering an engaging user experience. The performance metrics show that system latency is primarily influenced by the LLM-based code generation process rather than simulation execution. This latency could potentially be reduced by using more lightweight LLMs.

VI. Discussion

Based on our current implementation and initial feedback, we identify several key directions for future development:

1) **Enhanced Spatial Reasoning:** While our current LLM-based approach handles basic spatial relationships effectively, complex spatial reasoning remains challenging. Integrating specialized geometric reasoning modules and semantic scene graphs can potentially improve the system's understanding of spatial relationships.
2) **Complex Scenario Generation:** While we demonstrate scenes with some complexity for human-robot interaction, more complicated scenes can be challenging to generate, especially when there are dynamic components or a wide variety of objects. Current advancements in asset generation has the potential to narrow this gap.
3) **Sim-to-Real:** Like any simulation platform, RCareGen faces sim-to-real challenges. We envision leveraging more advanced sim-to-real pipelines to create digital twins that align more closely with the real world.
4) **Advanced Feedback Processing:** Building on our current feedback mechanism, we have identified the following ways to improve RCareGen:
   - Learning from user feedback patterns to improve system responses.
   - Supporting multi-modal feedback combining natural language, gestures, and visual indicators.
   - Developing personalized interaction models that adapt to individual user preferences.

By combining a web-based interface, LLM-based code generation, and robust simulation capabilities, RCareGen introduces a novel approach to democratizing robotics simulation and programming. Furthermore, its scalable and modular design ensures adaptability for emerging LLM technologies and evolving user needs. While there are still challenges to address, particularly in complex spatial reasoning and real-world implementation, the current system provides a foundation for more intuitive and accessible human-robot interaction development. As we continue to enhance the system's capabilities, we believe RCareGen will contribute to broadening participation in robotics research and development across diverse domains.

REFERENCES

[1] R. Ye, W. Xu, H. Fu, R. K. Jenamani, V. Nguyen, C. Lu, K. Dimitropoulou, and T. Bhattacharjee, "RCareWorld: A human-centric simulation world for caregiving robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 33–40, 2022.

[2] Z. Erickson, V. Gangaram, A. Kapusta, C. K. Liu, and C. C. Kemp, "Assistive gym: A physics simulation framework for assistive robotics," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[3] Y. Wang, Z. Sun, Z. Erickson, and D. Held, "One policy to dress them all: Learning to dress people with diverse poses and garments," in *Robotics: Science and Systems (RSS)*, 2023.

[4] OpenAI, "Gpt-3.5." https://platform.openai.com/docs/models/gpt-3-5, 2023. Accessed: 2025-01-12.

[5] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *arXiv preprint arXiv:2209.07753*, 2022.

[6] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11523–11530, 2023.

[7] 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," in *International Conference on Machine Learning (ICML)*, pp. 51936–51983, 2024.

[8] C. Li, F. Xia, R. Martín-Martín, M. Lingelbach, S. Srivastava, B. Shen, K. E. Vainio, C. Gokmen, G. Dharan, T. Jain, A. Kurenkov, K. Liu, H. Gweon, J. Wu, L. Fei-Fei, and S. Savarese, "igibson 2.0: Object-centric simulation for robot learning of everyday household tasks," in *Conference on Robot Learning (CoRL)*, pp. 455–465, 2022.

[9] "AI2-THOR: An Interactive 3D Environment for Visual AI," *ArXiv*, vol. abs/1712.05474, 2017.

[10] J. Hou, A. Dai, and M. Nießner, "3d-sis: 3d semantic instance segmentation of rgb-d scans," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[11] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, W. Ai, B. Martinez, *et al.*, "Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation," *arXiv preprint arXiv:2403.09227*, 2024.

[12] A. Abid, A. Abdalla, A. Abid, D. Khan, and M. Hraiz, "Gradio: Simplifying machine learning model interfaces." https://gradio.app, 2019.