Enabling LLM-based Environment Manipulation in Robotic Simulations via MCP

Gabriel Souza Borges
Universidade de Pernambuco
gabriel.sborges@upe.br

Bruno José Torres Fernandes
Universidade de Pernambuco
bruno.fernandes@poli.br

I. Introduction

The deployment of autonomous robots in dynamic, unstructured environments, such as homes and hospitals, remains a significant challenge for robotics [1]. Traditional programming is rigid and cannot anticipate the infinite variability of real-world scenarios. The rise of Large Language Models (LLMs) offers a promising solution, harnessing their advanced reasoning and natural language understanding capabilities to decompose complex tasks into executable steps [2].

However, a critical gap persists. LLMs are inherently "disembodied" and lack an innate understanding of the physical world. This lack of physical "grounding" often leads to action plans that are infeasible or unsafe. Although pioneering frameworks such as SayCan [3] have demonstrated the potential of grounding LLM-generated instructions in robotic capabilities, and related protocols for agent-aware control have been proposed [4], a standardized and scalable approach remains an open challenge. To overcome this limitation, a robust and standardized bridge between the LLM's abstract intelligence and the robot's physical capabilities is required.

In this paper, we propose a novel framework that utilizes the Model Context Protocol (MCP) [5] to fill this gap. MCP, an open standard, functions as a universal interface that allows any AI agent to communicate with any external tool or system in a uniform manner. Our contribution is to validate an architecture that employs MCP to enable an LLM to perceive and act directly within a robotic simulation environment, establishing a foundation for more complex autonomous tasks. Our findings confirm that this integration is not only viable but also effective for direct environment manipulation.

II. Methodology

The proposed architecture consists of three primary components that work in concert to translate high-level intent into simulated actions.

A. LLM-based Agent (MCP Client)

The "brain" of our system is a software agent that acts as an MCP Host. It receives a high-level objective in natural language from the user and queries an LLM to decompose this instruction into a logical sequence of actions by utilizing the capabilities exposed by the robotic environment.

B. Environment Capabilities Server (MCP Server)

The MCP Server acts as the "nervous system," exposing the functionalities of the simulation environment as a set of standardized ‘Tools‘. Based on our implementation, the server exposes a comprehensive set of capabilities for scene and object manipulation. The key tools validated in our tests include:

- **‘create_random_scene‘**: Initializes the environment with a set of objects.
- **‘load_object‘**: Creates a new object from a URDF file at a specified position.
- **‘set_object_color‘**: Modifies the visual properties of an existing object.
- **‘get_object_info‘**: Retrieves state information, such as position and orientation, for a specified object.

C. Simulated Environment (PyBullet)

We utilized PyBullet to create a physics-based simulation environment that represents the robot's "body" and "world." The MCP Server interacts directly with the PyBullet API to execute the actions requested by the MCP Client. The communication flow is cyclical: the LLM agent plans an action, the Client invokes the ‘Tool‘, the Server translates the call into a PyBullet API command, the simulation is updated, and the result is returned to inform the LLM's next step.

III. Results

To validate our framework, we conducted an interactive session demonstrating the LLM's ability to understand sequential, context-dependent commands to manipulate the simulation environment. The test confirms that the end-to-end communication pipeline is functional and effective.

The validation session proceeded as follows:

1) **Scene Initialization**: The user began with the command, "Load the room for tests." The LLM agent correctly interpreted this and invoked the ‘create_random_scene‘ tool, which populated the simulation with 10 random objects.
2) **Object Creation**: Next, the user commanded, "Create a cube at the center." The agent invoked the ‘load_object‘ tool, which successfully loaded a ‘cube.urdf‘ file into the simulation at position [0, 0, 0.5] and assigned it object ID 11.

3) **State Modification**: The user then gave a contextual command, ”Turn it green.” The agent correctly inferred that ”it” referred to the most recently created object (ID 11) and called the ‘set_object_color‘ tool, successfully changing the cube’s color.
4) **State Perception**: Finally, the user queried, ”What is the position of this cube?” The agent invoked the ‘get_object_info‘ tool for object 11. The server retrieved the state data from PyBullet and returned the correct position, which the agent relayed to the user.

This sequence demonstrates a complete perception-action loop. The system successfully translated a series of natural language commands into specific API calls, maintained context across turns, and used tools to both modify and query the state of the simulated world.

IV. Conclusion

In this work, we presented and validated a framework that successfully uses the Model Context Protocol to connect an LLM-based agent to a robotic simulation environment. Our results, based on direct, interactive tests, demonstrate that this architecture enables an LLM to create, modify, and query objects within a simulated world. This serves as a crucial proof-of-concept for the development of more autonomous and flexible robotic systems.

Having established the viability of the core integration, future work will focus on expanding the set of robotic ‘Tools‘ to include navigation, manipulator control, and object grasping. The ultimate goal is to chain these capabilities to enable the completion of complex, end-to-end tasks. Future research will also explore the transfer of the system from simulation to a physical robot platform and the performance of quantitative performance analyses.

References

[1] H. M. G. W. M. B. Ekanayake, D. G. K. Madusanka, M. A. V. J. Muthugala, and K. P. N. Jayasena, ”Challenges and Solutions for Autonomous Ground Robot Scene Understanding and Navigation in Unstructured Outdoor Environments: A Review,” *Applied Sciences*, vol. 13, no. 17, p. 9877, Aug. 2023.

[2] P. Li, Z. An, S. Abrar, and L. Zhou, ”Large Language Models for Multi-Robot Systems: A Survey,” 2025, arXiv:2502.03814.

[3] A. Ahn *et al.*, ”Do As I Can, Not As I Say: Grounding Language in Robotic Affordances,” 2022, arXiv:2204.01691.

[4] L. Lee and J. Lau, ”Robot Context Protocol (RCP): A Runtime-Agnostic Interface for Agent-Aware Robot Control,” 2025, arXiv:2506.11650.

[5] Anthropic, ”Model Context Protocol Specification,” modelcontextprotocol.io, 2024. [Online]. Available: https://modelcontextprotocol.io/specification/2025-06-18