A Study on the LLM-Based Software Architecture for Multi Robot Control

Sangho Choi

ABSTRACT

This paper examines the latest developments in the convergence of robotics and AI, with a particular focus on analyzing key cases of robot control research using large language models (LLMs). Recent AI technologies such as computer vision, neural networks, reinforcement learning, and LLMs have significantly improved robots' visual recognition, motion control, perception, and decision-making abilities. This study presents an LLM-based robot control architecture that can be applied to small, medium-sized, and home/educational robots without requiring large-scale robot data.

This architecture proposes a 'self-describing' method that automatically generates system prompts from robot control programs, allowing non-experts to easily control robots. Additionally, it offers various LLM model selection options to reduce API usage costs and presents a multi-agent architecture for future robot functionality expansion and control of multiple robots. The feasibility of the proposed method is verified through robots tests using a robot simulator. As robot popularization expands beyond manufacturing sites into everyday life, this research is expected to lower the barriers to applying AI technology to small and medium-sized robots.

☞ keyword : AI, Large Language Models, robot, control, sofrtware architectdure, few-shot learning, prompt

1. Introduction

Since the introduction of Unimate, the first industrial robot, at General Motors in 1961, robotics has driven innovative changes across various sectors, maximizing automation and efficiency. Recently, the convergence of artificial intelligence (AI) and robotics has further expanded its potential and effectiveness. Robots' capabilities have been significantly enhanced through the integration of multiple AI technologies: computer vision and neural networks have improved visual recognition capabilities, reinforcement learning has advanced motion control, and Large Language Models (LLMs) have enhanced cognitive and decision-making abilities.

AI technologies have brought transformative improvements to robots' three fundamental capabilities: perception, planning, and action. From a perceptual perspective, advancements in computer vision and sensor technologies have enabled more precise environmental recognition. In terms of planning capabilities, AI technologies, particularly reinforcement learning, have enabled efficient planning of complex tasks. The action control domain has seen substantial improvements in motion accuracy through sophisticated algorithms and real-time feedback systems.

The combination of LLMs exceptional capabilities and familiar chatbot interfaces has opened new frontiers in AI robotics research. LLMs, such as OpenAI's GPT-3, with their remarkable natural language processing capabilities, are being extensively studied in the domain of robot control. These advancements in LLMs have facilitated more intuitive and natural human-robot interaction, enabling simpler execution of complex task instructions and enhanced environmental understanding (Brown et al., 2020; Huang et al., 2022).

This paper examines a critical aspect of robot control: prompt generation methodology. Conventional approaches necessitated manual system prompt updates with each functional addition. To overcome this constraint, this study proposes an automated prompt generation methodology that leverages Python's decorator functionality to derive prompts directly from robot function definitions. Additionally, the research presents an architecture that accommodates multiple LLM engines and

introduces a framework for discriminative command execution across multiple entities (agents), including multi-robot systems.

The primary objective of this study is to enhance the efficiency and accuracy of LLM-based robot control while demonstrating its potential across various application domains. To achieve this, this paper proposes a methodology for validating and optimizing robot control outcomes using the PyBullet simulator. This approach offers the advantage of testing diverse scenarios while minimizing risks in real-world environments.

Furthermore, this study presents the design and implementation methodology for an efficient LLM-based robot control system. The organization of this paper comprises the following sections: theoretical foundations of robot control systems and LLMs; methodological framework for implementing LLM-based robot control systems; experimental evaluation and analysis of the proposed system's performance; and conclusions, including key findings and directions for future research.

This research contributes to the field by presenting a practical methodological framework for the design and implementation of LLM-based robot control systems, advancing robots' capabilities in perception, planning, and control. Furthermore, this study investigates the innovative potential applications emerging from the integration of robotics and AI, establishing groundwork for the development of more sophisticated and adaptable robotic systems.

2. Theoretical Background and Related Work

2.1 Overview of Robot Control Systems

Robot control systems serve as fundamental components for executing precise robotic operations through the acquisition and processing of sensor data to generate appropriate control commands. These systems are primarily classified into open-loop and closed-loop architectures.

Open-loop control systems generate outputs based on input signals without feedback mechanisms, providing architectural simplicity but limited environmental adaptability. Conversely, closed-loop control systems employ feedback loops for continuous operational adjustment to achieve desired states, exemplified by PID control, adaptive control, and predictive control methodologies.

The evolution of robot control technologies spans multiple technological epochs. While initial implementations predominantly employed open-loop control architectures, the advent of closed-loop control technologies in the 1960s-70s facilitated complex task execution. The 1980s witnessed the emergence of sophisticated control algorithms enabled by microprocessor advancement, while the 1990s marked the integration of artificial intelligence and machine learning, significantly enhancing autonomous capabilities and adaptive responses. Contemporary research emphasis has shifted toward deep learning for visual perception, reinforcement learning for autonomous control systems, and natural language processing for human-robot interaction interfaces (Javier et al., 2018; Liu et al., 2021). Additionally, research into collaborative control methodologies incorporating distributed control systems and multi-agent architectures has intensified, facilitating the expansion of robotic applications across diverse sectors, including industrial, medical, and service domains.

2.2 LLMs: Progress and Applications

LLMs constitute a breakthrough technology in Natural Language Processing (NLP), emerging from accelerated developments in artificial intelligence. LLM advancement has substantially enhanced capabilities in text comprehension and generation, demonstrating significant impact across diverse application domains.

The trajectory of artificial intelligence

commenced in the 1950s. Initial AI research originated from explorations into machine-based problem-solving through human-analogous cognitive processes. The term ‘artificial intelligence' was formally introduced at the 1956 Dartmouth Conference, leading to subsequent developments in expert systems and rule-based architectures (Rawas, 2024). The 1980s marked the advancement of neural network theory, establishing machine learning as the predominant paradigm in AI research.

The 1990s and early 2000s witnessed the ascendance of data mining and statistical learning methodologies, succeeded by the proliferation of deep learning in the 2010s. Deep learning advancement, leveraging extensive datasets and enhanced computational capabilities, achieved unprecedented performance in diverse domains, encompassing speech recognition, image classification, and natural language processing.

LLMs emerged as a response to fundamental limitations in natural language processing within this evolutionary trajectory of artificial intelligence. Initial NLP implementations relied predominantly on rule-based methodologies or statistical models trained on constrained datasets, exhibiting significant limitations in processing complex linguistic contextual relationships.

Natural Language Processing reached a pivotal turning point in June 2018 with OpenAI's introduction of GPT (Generative Pre-trained Transformer) (Radford et al., 2018). Subsequently, in October of the same year, Google introduced BERT (Bidirectional Encoder Representations from Transformers) (Delvin et al., 2019). BERT, based on the transformer architecture, demonstrated superior performance across various NLP tasks through its capability to comprehend context bidirectionally. The continuous evolution of OpenAI's GPT series further expanded the potential of LLMs. Notably, GPT-3, released in 2020, demonstrated human-comparable text generation capabilities through its 175 billion parameters.

LLMs exhibit several distinctive architectural characteristics. First, they utilize extensive pre-training on large-scale datasets, facilitating the acquisition of linguistic patterns and structural regularities through comprehensive textual corpus analysis. Second, they leverage parallel processing capabilities inherent to the transformer architecture, enabling efficient processing of extended contextual sequences and substantially accelerating learning and inference processes through parallel computational mechanisms. Third, they demonstrate zero-shot and few-shot learning proficiency, enabling task execution through general language understanding capabilities without task-specific fine-tuning procedures.

Contemporary advancements in LLMs are characterized by concurrent progression in model architecture scalability and performance metrics, coupled with application domain diversification. Multiple large-scale language models have emerged, including OpenAI's GPT-4, Google's PaLM (Pathways Language Model), and Meta's LLaMA, exhibiting superior performance across traditional NLP tasks, creative content generation, computational assistance, and complex problem resolution frameworks (Zhao et al., 2023).

Additionally, commercial implementations of LLMs are experiencing rapid proliferation. The integration of LLMs across diverse operational domains—encompassing conversational interfaces, machine translation systems, content generation platforms, intelligent virtual assistants, and automated customer service solutions—has yielded substantial improvements in operational efficiency and productivity metrics. Significantly, LLMs serve as fundamental enablers in the delivery of personalized user experience frameworks.

A significant new trend is emerging in recent LLM developments: the advancement of sLLMs (small Language Models) and On-Device AI. sLLMs represent technology

that maintains performance while reducing model size or optimizing for specific tasks, increasing their utility in mobile devices and edge computing environments. Concurrent developments in On-Device AI technology provide benefits including enhanced privacy protection and reduced network latency. These technologies are expanding the application scope of LLMs and are expected to play crucial roles in systems requiring real-time processing, such as robot control systems and mobile applications (Yi et al., 2023). The progression of sLLMs and On-Device AI facilitates more efficient and widespread implementation of LLM technologies.

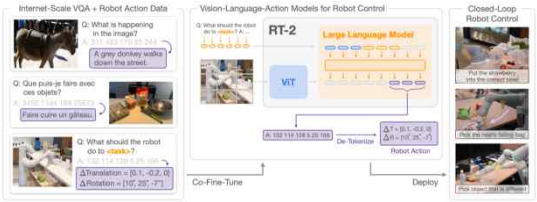
In conclusion, the advancement of Large Language Models continues to reveal unprecedented possibilities in artificial intelligence, with projected significant implications across NLP and diverse application domains. This research investigates the operational efficiency and practical utility of distributed multi-agent architectures through the implementation of LLM characteristics in robotic control systems. The integration of LLM's natural language processing capabilities with robotic systems is anticipated to facilitate the development of more sophisticated and adaptable architectural frameworks.

2.3 Multi-modal Approaches in Robot Control

The convergence of robotics with various domains of AI continues to accelerate, with multimodal approaches significantly enhancing robotic intelligence and behavior. Multimodal robot control enables more sophisticated and autonomous robotic operations through simultaneous processing of various data modalities, including vision, audio, and text.

2.3.1 Robot-Vision Integration

The evolution of computer vision technologies is fundamentally transforming robotic environmental perception and interaction paradigms. Google's recently introduced RT-2 (Vision-Language-Action Models) demonstrates novel methodologies in robotic control through the synthesis of vision and language models. RT-2 facilitates enhanced operational intuition and behavioral flexibility by integrating visual data streams with language model architectures, enabling sophisticated robot-environment interactions (Brohan et al., 2023).



(Figure 1) Robot Control using Vision-Language Model Capabilities

RT-2 demonstrates three fundamental architectural characteristics. First, the integration of vision and language modalities enables robotic systems to parse and comprehend visual data through natural language frameworks, facilitating enhanced interpretation of complex visual environments and precise execution of user-specified directives. Second, RT-2 implements a transformer-based Vision-Language Model (VLM) architecture for concurrent processing of visual and linguistic data streams, utilizing this integrated analysis for action determination. Third, the model exhibits zero-shot generalization capabilities through extensive training on heterogeneous datasets, enabling adaptive responses to novel operational contexts and environmental conditions not explicitly encountered during the training phase.

2.3.2 Robot-LLMs Integraion

LLMs enable enhanced human-robot interaction paradigms through advanced natural language processing capabilities. LLMs facilitate robot comprehension and execution of complex directives by leveraging linguistic understanding and generation mechanisms derived from textual corpus analysis. This functionality,

when integrated with visual perception systems, enables more sophisticated and adaptable robotic behavioral patterns.

Robotic control systems incorporating LLMs have emerged as a significant research focus. ProgPrompt represents an innovative implementation utilizing large language models for generating robotic task planning sequences (Singh et al., 2022). This research framework enables the derivation of robotic task plans from natural language specifications, facilitating user-directed complex task execution. While ProgPrompt demonstrates substantial improvements in task planning efficiency and adaptability, inherent linguistic ambiguities present limitations, potentially resulting in sub-optimal plan generation and execution precision.

SayCan study presented a methodology for robots to comprehend natural language commands and translate them into executable actions (Ahn et al., 2022). This research enhanced robots' ability to understand environmentally feasible actions and convert natural language commands into corresponding behaviors. While limitations remain in precise action determination within complex environments considering all possibilities, the study's significance lies in its advancement of robots' capabilities to accurately interpret and execute user commands.

Code as Policies study proposed a methodology for generating robot control programs using language models (Liang et al., 2023). This research demonstrated LLMs' capability to comprehend programming languages and generate robot control code, validating their potential to produce code solutions for complex robotic control problems. However, the study indicated limitations regarding the execution efficiency and operational stability of generated code in real-world environments.

Research has explored the potential applications of large language models like ChatGPT in robot control systems (Vemprala et al., 2023). This study established design principles focusing on natural language interaction, situational adaptability, and safety assurance, demonstrating LLMs' effectiveness in robot control through their capabilities in natural language comprehension, situational awareness, and adaptive learning. While the research validated LLMs' ability to accurately and naturally interpret user commands, limitations persist regarding comprehensive understanding and adaptation across all operational scenarios.

(Figure 2) Robot control with chatGPT

2.3.3 Multimodal Robot Control

Multimodal robotic control systems demonstrate significant advantages over unimodal approaches. First, they enhance environmental perception capabilities through the integration of heterogeneous data modalities. For instance, the synthesis of visual and linguistic data streams enables more precise command interpretation based on visual perception frameworks. Second, multimodal data integration provides diverse analytical perspectives in decision-making processes, facilitating more sophisticated and adaptable behavioral responses. Third, the system enhances operational utility through more naturalistic and intuitive human-robot interaction paradigms.

Contemporary research in multimodal robotic control systems is advancing toward optimization of autonomous capabilities and environmental adaptability through the synthesis of diverse artificial intelligence technologies. For instance, intensive investigation is being conducted on the integration of LLMs with reinforcement learning frameworks to facilitate complex task acquisition and execution. Furthermore, technological developments

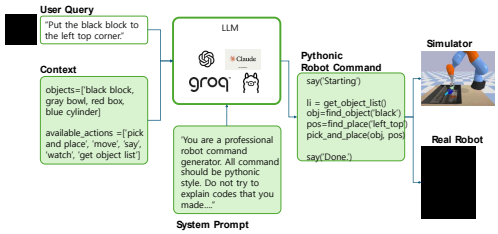
incorporating vision systems with LLMs are enabling real-time environmental analysis and appropriate action execution based on natural language command interpretation.

In conclusion, multimodal robotic control architectures are substantially advancing robotic cognitive capabilities and behavioral competencies, with LLM-based control systems playing an instrumental role in enabling autonomous operation within complex operational environments. These research initiatives are facilitating the expansion of robotic implementation domains and enhancing deployment feasibility across diverse industrial sectors. Future research trajectories are anticipated to focus on addressing the constraints of current multimodal methodologies and developing more robust and computationally efficient robotic control architectures.

3. Research Methodology

3.1 Design of LLM-based Control Module

For robots to comprehend operators' commands delivered in natural language, several LLM application techniques exist. This study implements few-shot learning prompting techniques as illustrated in Figure 3. Few-shot learning enables rapid learning and generalization from a limited number of examples, proving particularly valuable in robotic control where pre-training for all possible scenarios is infeasible. Combined with LLMs' robust generalization capabilities, few-shot learning enables more flexible and adaptive robot operation, which is particularly crucial when deploying robots in diverse and unpredictable real-world scenarios.



(Figure 3) Conceptual Framework of Robot Command Generation using LLM

Within the control module architecture, an LLM Selector component, engineered for LLM selection and management utilizing LangChain, implements chaining operations through LCEL (LangChain Expression Language). This architectural framework enhances system flexibility and extensibility through dynamic integration of decomposed components: prompt structures, LLM engine implementations, and output parsing mechanisms.

Furthermore, for automated prompt generation in robotic command synthesis, the system implements a methodology leveraging Python's decorator functionality to programmatically extract Agent class methods and dynamically integrate them into few-shot learning prompt structures. The '@include_action' decorator specification facilitates automatic integration of method annotations into prompt compositions, thereby enhancing architectural flexibility during functional expansion implementations.

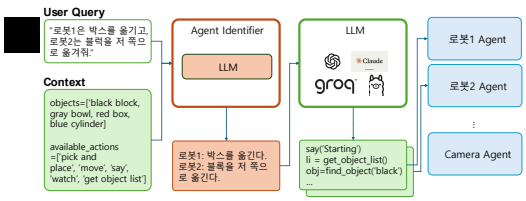
```python
# Custom Decrorator
# 이 데코레이터가 선언된 메서드만 프롬프트(Context)에 자동으로 포함된다.
# LLM은 Context에 정의된 메서드들만을 로봇명령어 생성에 사용할 수 있다.
def include_action(func):
    func._include_action = True
    return func
```

```python
@include_action
@debug_codeExecution
def get_camera_image(self, showImage=True):
    """
        # Take some pictures
        say(f'OK - taking some pictures')
        get_camera_image(showImage=True)
    """
    # Compute view and projection matrices
    view_matrix = p.computeViewMatrix(
        self.cameraPostion, [0, 0, 0], [0, 0, 1])
    projection_matrix = p.computeProjectionMatrixFOV(
        self.fov, self.aspect, self.near, self.far)

    # Capture the image
    images = p.getCameraImage(
        self.width, self.height, view_matrix, projection_matrix,
        shadow=True, renderer=p.ER_BULLET_HARDWARE_OPENGL)
    # images[0] : width
    # images[1] : height
    # images[2] : RGB pixel data
    # images[3] : depth data
    # images[4] : segmentation mask
```

(Figure 4) Automated Prompt Generation using Decorator Implementation

The system implements a mechanism enabling LLM to identify individual agents in multi-robot scenarios. For example, when given a user command "Robot1, move the box, and Robot2, move the block over there.", the LLM engine redefines commands in an "{agent}:{action}" format, allowing each agent to execute its designated task. This implementation enables effective control in complex multi-robot environments.



(Figure 5) Identification of Multi Agent

3.2 Robot Simulator

Given the proximity of robotic operational environments to human spaces, simulation-based motion validation is essential prior to physical deployment. This study conducted comparative analysis of various robot simulators, ultimately selecting PyBullet for implementation. PyBullet offers advantages including straightforward installation, support for diverse robotic models, robust physics engine capabilities, open-source accessibility, real-time simulation functionality, and an active user community.

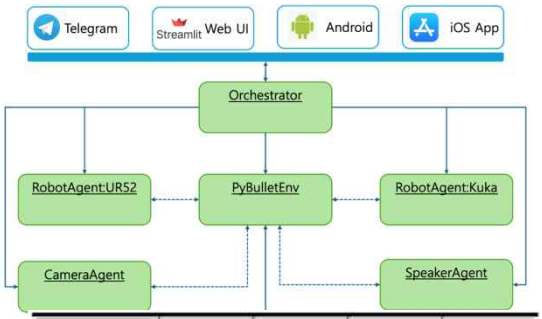
(Table 1) Types of Robot Simulators

Functions	PyBullet	Gazebo	MuJoCo	Webots
Language	Python	C++, Python	Python, C	C++, Python, Matlab, Java
Usage	High	Moderate	Moderate	Moderate
Accuracy	Moderate	High	High	High
Format	URDF, SDF, MJCF	URDF, SDF	MJCF	URDF, VRML
Real Time	Yes	Yes	Yes	Yes
Open Source	Yes	Yes	No	Yes
Performance	Moderate	High	Very High	High
Community	High	High	Moderate	Moderate

The PyBullet simulation environment served as an instrumental platform for performance validation and optimization of the proposed LLM-based robotic control architecture, facilitating the establishment of operational safety protocols and efficiency metrics prior to physical system implementation.

3.3 System Architecture

The system architecture proposed in this investigation is structured as a collaborative multi-agent framework, facilitating coordinated task execution among multiple robotic agents and supplementary agent entities. The fundamental architectural components encompass the Orchestrator for system coordination, PyBulletEnv for environmental simulation, and diverse agent implementations for task execution.



(Figure 6) Overall System Structure

The 'Orchestrator' module functions as the central control hub of the system. It manages communications between external interfaces and internal agents, processing user natural language commands for internal agent distribution and returning results to external interfaces. Additionally, it generates and manages the PyBullet-based main simulation environment (PyBulletEnv).

'PyBulletEnv' serves as a PyBullet-based simulation environment that simulates the operations of various robots and objects. It executes simulations based on commands received from the Orchestrator and interacts with the objectManager to generate and

manage objects within the simulation.

Various agents perform specific roles within the system. The RobotAgent controls robots such as UR5e and Kuka, utilizing the LLM engine to convert natural language commands into executable robot commands. The CameraAgent and SpeakerAgent are responsible for collecting visual information from the simulation environment and managing audio output, respectively.

Individual agents execute domain-specific functionalities while the Orchestrator facilitates comprehensive system integration. PyBulletEnv provides high-fidelity physical simulation capabilities, enabling pre-deployment validation and operational optimization of robotic movements.

The implemented architectural framework facilitates natural language interface for complex robotic control directives, with the system architecture enabling efficient command processing for safe and precise robotic operation execution. Additionally, the distributed architectural paradigm ensures enhanced system scalability and operational flexibility, establishing platform-agnostic compatibility across diverse robotic systems and environmental contexts.

4. Experimental Results

4.1 Comparative Analysis of LLM Code Generation Performance

While LLMs conventionally generate code accompanied by comprehensive explanations in response to natural language queries, robotic control applications require exclusively executable code output. Consequently, system prompts are implemented to explicitly define LLM operational parameters and constraints. To validate the efficacy of robotic command code generation, computational latency and code generation accuracy are evaluated across multiple LLM implementations using four standardized natural language test cases, as illustrated in Figure 7.

(Figure 7) Questions for LLMs' Code geration Evaluation

Performance analysis indicates that OpenAI (model: gpt-3.5-turbo-0125) achieves optimal accuracy (100%) while maintaining comparatively efficient computational latency. Groq exhibits minimal execution latency but demonstrates suboptimal accuracy metrics. LLAMA (model: Codellama:7b) maintains moderate computational efficiency but exhibits reduced accuracy performance (50%), while Claude (model: claude-3-opus-20240229) demonstrates maximal accuracy (100%) but requires substantially increased processing duration.

(Table 2) LLM Execution Time(s)

LLM	Question (1)	Question (2)	Question (3)	Question (4)
OpenAI	0.59	0.87	0.91	1.41
Groq	0.39	0.49	0.35	0.41
LLAMA	4.75	1.24	1.92	1.39
Claude	2.80	1.93	5.98	3.61

(Table 3) LLM Accuracy Summary(%)

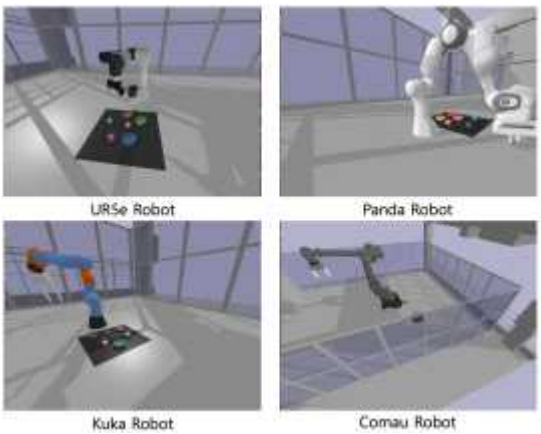
LLM	Question (1)	Question (2)	Question (3)	Question (4)
OpenAI	100	100	100	100
Groq	100	75	75	50
Codellama:7b	50	100	50	50
Claude	100	100	100	100

Although open-source models currently exhibit lower accuracy metrics compared to proprietary implementations, their accelerated development trajectory and enhancement rate necessitate periodic performance assessments. This research implements OpenAI's model as the primary LLM engine for robotic control applications, selected from among the highest-accuracy

performers (OpenAI and Claude), based on its superior computational efficiency metrics.

4.2 Robot control Experiments

To validate LLM control implementation across heterogeneous robotic platforms, a test environment was configured with four distinct robotic systems, incorporating both mechanical gripper and vacuum-based end effector configurations.



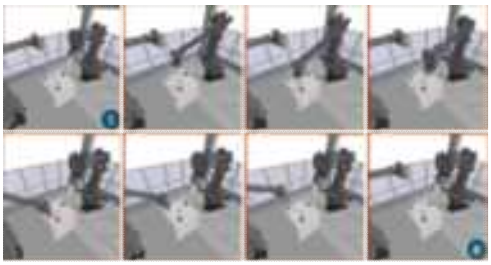
(Figure 8) Robots for simulation

The system uses Telegram as the command transmission protocol interface. User-initiated Telegram string inputs are processed through the Orchestrator, where natural language instructions undergo LLM-based analysis and transformation into pythonic robot command structures, culminating in physical task execution by the robotic system.

The experimental validation protocol was executed in two phases: initial evaluation of natural language command interpretation for fundamental operations on a single robotic platform, followed by sequential operation validation across multiple robotic systems. The implemented natural language test commands were as follows:

"I want robot1 to put the red block on the same-colored bowl, and robot2 will move the orange block on the gray bowl."

Operational verification demonstrated successful command differentiation and execution by both robotic platforms (robot1 and robot2), with movement sequences conforming to operator-intended trajectories and task specifications. (https://youtu.be/Gg9zhdg3zmg)



(Figure 9) Sequential Control of multi robots

5. Conclusion and Future Research Directions

This research aimed to design and implement a multi-agent architecture for LLM-based robot control, validating its performance through diverse experimental evaluations. Primary research achievements include establishing an environment for selecting and utilizing various LLM engines through LLM Selector design. System flexibility and scalability were significantly enhanced by implementing LangChain to separate and dynamically link prompts, LLM engines, and output parsers.

Furthermore, an automated prompt generation method utilizing Python decorators enabled automatic system prompt updates with each functionality addition, substantially improving user and developer convenience. Additionally, the implementation of a multi-agent identification mechanism enabled precise command recognition and execution by individual agents in multi-robot environments.

Finally, system practicality and stability were validated through experiments utilizing the PyBullet simulator. Experimental results

demonstrated high accuracy and efficiency, with superior code generation capabilities and robot control performance achieved through diverse LLM model implementations.

Primary limitations and future research directions of this study include insufficient validation across diverse environments, necessitating more rigorous verification of real-world applicability. Additionally, further research is required for precise control in complex command scenarios and exceptional situations due to inherent LLM model limitations.

System scalability presents significant challenges, requiring solutions for network latency, resource management, and synchronization issues that may arise during large-scale deployment. Limitations in automated prompt generation necessitate additional research for accurate prompt creation in cases involving complex functions and exception handling logic. Finally, improvements in user interface and interaction are required, with enhanced UI/UX needed to facilitate more intuitive and accessible system operation for end users.

Future research necessitates exploration of diverse methodologies to address these limitations and enhance system practicality. Key research challenges include implementation of sLLM and On-Device AI in real environments, ensuring operational safety regarding inter-robot motion interference, advancement of LLM models for enhanced environmental affordance and grounding, strengthening system scalability, improving automated prompt generation accuracy, and enhancing user interface design.

This research has presented practical methodologies for designing and implementing LLM-based robot control systems, contributing to the advancement of robotic perception, planning, and control capabilities. Future investigations are expected to enable the development of more sophisticated robotic control systems.

Reference

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. and Amodei, D. (2020), Language models are few-shot learners, *Advances in Neural Information Processing Systems, 33, 1877-1901.*

Huang, W., Abbeel, P., Pathak, D., & Mordatch, I. (2022), Language models as zero-shot planners: extracting actionable knowledge for embodied agents. Proceedings of the 39th International Conference on Machine Learning, 9383-9407.

Javier Ruiz-del-Solar, et al. (2018), A Survey on Deep Learning Methods for Robot Vision, *arXiv.* *https://arxiv.org/abs/1803.10862*.

Liu, R., et al. (2021), Deep Reinforcement Learning for the Control of Robotic Manipulation: A Focused Mini-Review, *arXiv:2102.04148.*

Rawas, S. (2024), AI: the Future of Humanity. Discover Artificial Intelligence. 4(25), *Springer*

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018), Improving language understanding by generative pre-training. *OpenAI Technical Report.*

Devlin, J., et al. (2019), BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805*.

Zhao, W. X. and Zhou, K., et al. (2023), A survey of Large Language Models. *arXiv:2303.18223*

Yi, L., Guo, L., Zhou, A., Wang, S., Xu, M. (2023), EdgeMoE: Fast on-device inference of MoE-based large language models, *arXiv:2308.14352.*

Signh,I., Blukis, V., et al. (2023), ProgPrompt: Generating Situated Robot Task Plans using Large Language Models, *In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), 11686-11693.*

Ahn, M., Brohan, A., Brown, N., et al. (2022), Do as I can, not as I say: grounding language in robotic affordances, *arXiv:2204.01691.*

Liang, Y., Cabi, S., et al. (2023), Code as Policies: Language Model Programs for Embodied Control, *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 6045-6056.*

Vemprala, S., Bonatti, R., Bucker, A., Kapoor, A. (2023), ChatGPT for Robotics: Design Principles and Model Abilities, *arXiv:2306.17582.*

◐ ABOUT THE AUTHOR ◑

Sangho Choi

Enrolled in the Master's in AI·Big Data program at the Graduate School of AI, Seoul School of Integrated Sciences & Technologies (aSSIST)

Areas of interest: AI, robot, LLM, Computer Vision, etc.

E-mail : sangho.0529@gmail.com