A Methodology for Real-Time Action Verification in Physical AI by Interpreting Task Outcomes

Soyeon Kim [a,b†], Jaejun Lee [a,b†], Dongju Kim [a], Yohan Lee [a], Hogeon Seo [a,b*]
[a]*Korea Atomic Energy Research Institute, 111, Daedeok-daero 989 beon-gil, Yuseong-gu, Daejeon, 34057, Korea*
[b]*University of Science & Technology, 217, Gajeong-ro, Yuseong-gu, Daejeon, 34113, Korea*
[*]*Corresponding author: hogeony@hogeony.com*

Keywords: Physical AI, Large Language Models, Action Verification

1. Introduction

Nuclear power stations are regarded as extreme industrial environments, where high reliability and safety are of the utmost importance. This situation can be attributed to the presence of hazards such as ionizing radiation, elevated temperatures, and complex structures that significantly restrict human access [1]. In situations where immediate action is required, such as during emergencies or critical equipment inspections, robotic systems that facilitate remote monitoring and manipulation are imperative for ensuring the safety of the operators. Furthermore, the integration of robots capable of operating in such conditions directly enhances the security and operational stability of nuclear facilities [2].

However, conventional robot control methods require intricate programming and precise manipulation skills, posing significant challenges for rapid perception and adaptive response to unpredictable changes [3]. These challenges are further compounded in atypical situations where pre-programmed responses may be inadequate or entirely absent, necessitating the development of novel control strategies by operators under time-critical conditions.

In order to address these limitations, control paradigms powered by large language models (LLMs) have emerged as a promising solution. LLMs have been demonstrated to possess the capacity to interpret highly abstract and complex natural language commands and to transform them into coherent robotic action sequences. This facilitates intuitive and flexible task definition and execution that was previously challenging [4,5]. This transition towards Physical AI, a paradigm that facilitates direct interaction between AI systems and the physical world, signifies a paradigm shift from purely computational intelligence to embodied intelligence capable of real-time physical manipulation and environmental adaptation [6].

The objective of this research is to evaluate the role of LLM-centric control in advancing intelligent robotic applications for nuclear contexts, and to verify its effectiveness as an integral module within a broader Physical AI framework for real-time interpretation and validation of robot behaviors in mission-critical scenarios.

2. Methods and Results

This section details the proposed methodology and system architecture for text-driven quadruped control. Section 2.1 outlines the hardware and software platform, while Section 2.2 introduces the lightweight LLM backend. Subsequently, Section 2.3 describes the end-to-end execution sequence. The single-command control flow is then validated in Section 2.4. Finally, Section 2.5 presents an experimental evaluation of the system's robustness against sequential commands.

2.1 Platform and SDK

We implement a text-to-motion pipeline for the Unitree Go2 using the Python interface to Unitree SDK2 [7]**.** The text-to-motion pipeline is initiated by a console command issued by the operator. The LLM control backend processes this command by utilizing its function calling capabilities to select an appropriate motion function and determine its parameters. Subsequently, a function executor module then maps this structured output to the required SDK command format, which the SDK then transmits to the robot for physical execution. A key architectural principle is the decoupling of the LLM from the high-frequency control loop. The LLM's role is strictly limited to high-level command interpretation via function calling, which effectively modularizes the text-based control logic. This design choice yields two significant advantages: it minimizes latency by shortening the command-to-actuation latency and ensures reproducibility by having the LLM trigger consistent, pre-defined robot behaviors through this modular interface.

2.2 LLM Backend

Our LLM backend is built upon Qwen3-4B [8], a model specifically chosen for its proficiency in tool and function calling. The model's primary role is to process a user's natural language command. It first performs initial interpretation of the user's intent. It then leverages its function calling mechanism to select the appropriate motion primitive from a predefined set of motion primitives and generates the required motion parameters.

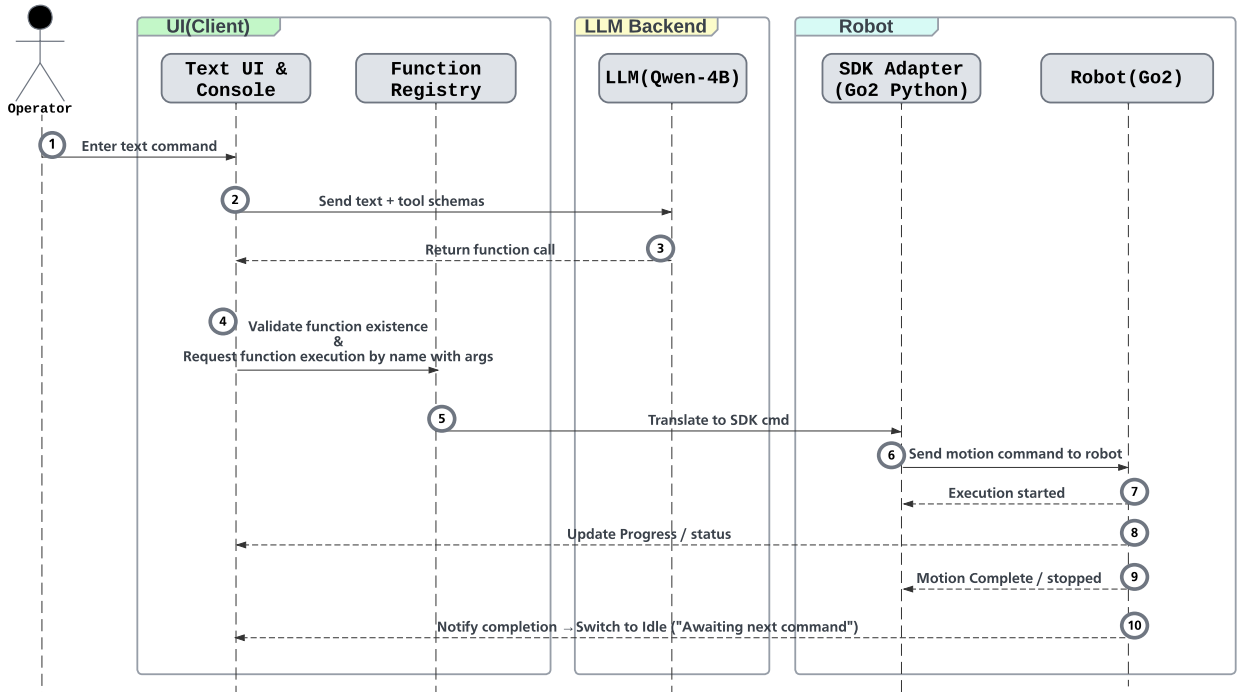

Fig. 1. Overview of the end-to-end text-to-motion execution sequence.

The resulting output is a structured function call, which is then passed to a client-side Function Registry.

This registry maps the call to the low-level SDK actuation commands required for robot actuation. For practical deployment, we utilize a quantized version of the lightweight 4B model on an 8 GB edge device. This configuration is viable as compact Qwen3 models are known for their computational efficiency under 4-bit or FP8 quantization, enabling real-time inference in resource-constrained environments.

2.3 End-to-End Execution Flow

The end-to-end text-to-motion sequence is orchestrated across a three-tiered architecture consisting of a UI client, an LLM backend, and the SDK/robot hardware, as depicted in Fig 1. The sequence is initiated when an operator inputs a natural language command via the console interface. This input, along with contextual information such as the outcome of the previous action and the current operational mode, is transmitted to the LLM backend. The LLM, in turn, generates a structured function call that specifies a motion primitive and its corresponding arguments. Within the UI client, a Function Registry is responsible for processing this call. Its tasks include parsing the function call, checking and adjusting arguments to ensure safe operational bounds, and finally, translating them into the native SDK command format. The formatted command is then dispatched to the quadruped via the SDK for physical actuation. Concurrently, the SDK streams real-time telemetry—including odometry, IMU data, and system status—back to the UI client, enabling continuous progress monitoring. Upon completion of the motion, the system generates an execution summary detailing the original command, total execution time, and termination state (e.g., success, failure). This summary is logged for diagnostic purposes and also provided as context to the LLM for subsequent interactions. Finally, the system transitions to an Idle state, awaiting the next user command.

2.4 Control Flow for a Single Command

Table I: Experimental validation of LLM-based Physical AI control system, demonstrating Unitree Go2 quadruped robot executing natural language commands: (a) idle state, (b) forward locomotion, (c) safety positioning, and (d) situational analysis.

			
(a)	(b)	(c)	(d)
Idle State	Go Forward	Lie Down	Explain the Situation

We validated the text-to-motion pipeline on a Unitree Go2 with four single-command primitives: Idle**,** Go forward**,** Lie down**,** and Explain the situation**.** The console UI sent each natural-language command to the Qwen3-4B backend, which returned a structured function call executed by the SDK. Table I lists these individually executable, language-driven actions, which serve as the representative capability set: panel (a) shows a zero-velocity overlay confirming the idle/ready state;

panel (b) shows forward-motion cues indicating straight locomotion; panel (c) shows a before/after posture pair with a small roll–pitch angle inset converging to the prone safety posture; panel (d) depicts the robot remaining stationary with a no-motion cue, denoting that the "Explain the situation" command produces a textual explanation but issues no motion (the text output is omitted from the figure). Together, Table I and Table II document the feasibility of the one-command control flow from intent parsing to actuation, completion reporting, and return to idle.

2.5 Experimental Results

2.5.1 Sequence-Length Robustness

Following the validation of single commands, we evaluated the system's robustness when executing sequential actions. We composed sequences from three motion primitives (Move, Lie down, Stand up) on a flat indoor floor and conducted 20 trials for each sequence length from one to five. We designed five test sequences with increasing complexity. The simplest sequence consisted of a single move command. This was followed by a sequence of two consecutive move commands, and a three-action sequence that included a lie down command. The complexity was further increased with a four-action sequence comprising two moves, a lie down, and then a stand up. The final five-action sequence involved moving three times before performing the lie down and stand up actions. Notably, each sequence was triggered by a single, high-level language command, such as "Move forward twice and lie down" for the three-action sequence, or "Move forward three times, then lie down and stand up" for the five-action sequence.

Table II: Success Rates as a Function of Sequential Command Length

Command	#Actions	Success/Trials	Success Rate [%]
C1: move x 1	1	19/20	95
C2: move x 2	2	18/20	90
C3: move x 2 → lie down	3	13/20	65
C4: move x 2 → lie down → stand up	4	7/20	35
C5: move x 3 → lie down → stand up	5	0/20	0

As summarized in Table II, the system maintained high reliability for sequences of one to two actions, achieving success rates of 95% (19/20) and 90% (18/20), respectively. However, performance began to degrade with three-action sequences, dropping to 65% (13/20). A sharp decline was observed for four-action sequences, which succeeded only 35% of the trials (7/20), and all five-action sequences failed (0/20). These outcomes indicate that the current pipeline reliably supports sequences of up to two or three actions under our test conditions. The degradation in performance with longer sequences is likely attributable to the accumulation of state estimation errors and minor physical instabilities between discrete actions. Without a mechanism to recalibrate or correct cumulative deviations, failure probability increases with each additional action.

3. Conclusions

The study validated the feasibility of an LLM-based Physical AI framework for intuitive, text-driven control of quadruped robots in complex industrial environments, such as nuclear facilities. Through comprehensive experimental validation using the Unitree Go2 platform, the system successfully interpreted and executed diverse natural language commands, confirming reliable command-to-action conversion and adaptive response behaviors. The efficacy of the framework is attributed to its hierarchical control architecture, which modularizes high-level command interpretation and low-level motion execution. This architecture facilitates the framework's capacity to achieve real-time performance and reproducibility, even in environments characterized by computationally constrained conditions. The findings indicate that LLM-based physical intelligence has the capacity to enhance safety and enable flexible task execution for autonomous robots in high-risk, mission-critical domains. Future research will focus on the enhancement of the framework's autonomy, robustness, and adaptability to dynamic and atypical situations. This will facilitate the reliable deployment of intelligent robots in nuclear and similarly extreme environments.

ACKNOWLEDGMENTS

This work was supported in part by Korea Atomic Energy Research Institute R&D Program grant (KAERI-524540-25), in part by the National Research Council of Science & Technology (NST) grant by the Korea government (MSIT) (No. GTL24031-000), and in part by the National Research Foundation of Korea grant by the Korean government (MSIT) (RS-2023-00253853).

REFERENCES

[1] W. Zhou, C. Zhang, S. Huang, Z. Wu, S. T. Revankar, "Reactor Fuels, Materials and Systems Under Extreme Environments", Frontiers in Energy Research, Vol. 10, pp. 860553, 2022.

[2] B. Bird, A. Griffiths, H. Martin, E. Codres, J. Jones, A. Stancu, B. Lennox, S. Watson, X. Poteau, "A robot to monitor nuclear facilities: Using autonomous radiation-monitoring

assistance to reduce risk and cost", IEEE Robotics \& Automation Magazine, Vol. 26, No. 1, pp. 35-43, 2018.
[3] S. Y. Park, C. Lee, S. Jeong, J. Lee, D. Kim, Y. Jang, W. Seol, H. Kim, S. H. Ahn, "Digital twin and deep reinforcement learning-driven robotic automation system for confined workspaces: A nozzle dam replacement case study in nuclear power plants", International Journal of Precision Engineering and Manufacturing-Green Technology, Vol. 11, No. 3, pp. 939-962, 2024.
[4] J. P. Espada, S. Y. Qiu, R. G. Crespo, J. L. Carus, "Leveraging large language models for autonomous robotic mapping and navigation", International Journal of Advanced Robotic Systems, Vol. 22, No. 2, pp. 17298806251325965, 2025.
[5] Y. Wang, Q. Wang, J. Fan, "High-Precision Control of Humanoid Muscle-skeleton Robotic Arm Using Reinforcement Learning and Large Language Models", IECON 2024-50th Annual Conference of the IEEE Industrial Electronics Society, pp. 1-6, 2024.
[6] R. S. Dewi, A. N. Kawakib, M. N. Laili, A. L. Fauziah, S. R. Sabrina, R. L. Hana, "A Systematic Review of Physical Artificial Intelligence (Physical AI): Concepts, Applications, Challenges, and Future Directions", Journal of Artificial Intelligence and Engineering Applications (JAIEA), Vol. 4, No. 3, pp. 2246-2253, 2025.
[7] Unitree Robotics, "unitree_sdk2_python: Python interface for Unitree SDK2," *GitHub repository*, 2025. Available: https://github.com/unitreerobotics/unitree_sdk2_python. (Accessed: Aug. 19, 2025)
[8] Qwen Team, "Qwen3-4B," *Hugging Face* (Model card), Aug. 6, 2025. Available: https://huggingface.co/Qwen/Qwen3-4B. (Accessed: Aug. 19, 2025)