Sensors and Materials, Vol. 37, No. 7 (2025) 3073–3089
MYU Tokyo

S & M 4104

LA-RCS: LLM-agent-based Robot Control System

Taek-Hyun Park,[1†] Young-Jun Choi,[1†] Seung-Hoon Shin,[1†]
Chang-Eun Lee,[2] and Kwangil Lee[1,3*]

[1]Department of Artificial Intelligence Engineering, National Korea Maritime & Ocean University, Busan, Korea
[2]Depense & Safety ICT Research Department, ETRI, Daejeon, Korea
[3]Department of Artificial Intelligence Research, DMASTA, Busan, Korea

(Received December 24, 2024; accepted June 27, 2025)

Keywords: LLM agent, robot task planning, AI

The large language model (LLM)-agent-based robot control system (LA-RCS) is a sophisticated robot control system designed to autonomously plan, work, and analyze the external environment on the basis of user requirements by utilizing LLM agents. Utilizing a dual-agent framework, LA-RCS generates plans on the basis of user requests, observes the external environment, executes the plans, and modifies the plans as needed to adapt to changes in external conditions. Additionally, LA-RCS interprets natural language commands by the user and converts them into commands compatible with the robot interface so that the robot can execute tasks and meet user requests properly. During this process, the system autonomously evaluates observation results, provides feedback on the tasks, and executes commands on the basis of real-time environmental monitoring, significantly reducing the need for user intervention in fulfilling requests. We categorized the scenarios that LA-RCS is designed to handle into four distinct types and subsequently conducted a quantitative evaluation of its performance across each scenario. The results showed an average success rate of 90%, demonstrating the system's capability to fulfill user requests satisfactorily. For more extensive results, readers can visit our project website: https://la-rcs.github.io.

1. Introduction

The emergence of large language models (LLMs) has shown their potential in the fields of complex problem solving, multi-stage inference, and natural language processing, especially with regard to generalization performance outcomes in various tasks(1–4) and ensuring excellent performance in natural language processing tasks such as understanding, generating, and translating text.(5–9) As the size of models increases, LLMs evolve to become closer to human thinking, executing instructions more accurately and consistently in response to user requests.(10)

Along with the expansion of LLMs, there has been research on visual LLMs (VLMs).(11–14) VLMs simultaneously learn the correlation between image and text data through a combination of vision models and language models, providing enhanced multimodality in response to user requests through their proficient processing capabilities for text and image data.(12,15,16)

*Corresponding author: e-mail: leeki@kmou.ac.kr
†These authors contributed equally to this work and should be considered co-first authors.
https://doi.org/10.18494/SAM5643

ISSN 0914-4935 © MYU K.K.
https://myukk.org/

Multimodal VLMs observe the user interface or graphical user interface (GUI) on the basis of image interpretation capabilities and efficiently perform tasks related to complex user requests and undertake repetitive tasks expressed in natural language using visual information through user interactions.(17)

Advancements in multimodal VLMs and related technologies have led to the emergence of LLM-based agents capable of providing problem-solving capabilities across a diverse range of environments.(18,19) LLM agents possess the ability to make intelligent decisions, enabling them to plan and reason effectively in response to natural-language-based requests.(20–23) These technological advances facilitate and provide an important foundation for the evolution of LLMs into large action models (LAMs). LAMs control other systems through communication with users and interact with the real world through pthe hysical behavior of the systems to meet user requests.

Previous research in the field of robot control required a large amount of learning data to operate robot systems according to user requests.(24) In addition, there are limitations (e.g., complexity and human intervention errors) when attempting to have robot systems make arbitrarily automated judgments to fulfill user requests with versatile environments,(25,26) although there have been endeavors to resolve these limitations.(27–29) To fulfill user requests, establishing and executing an appropriate plan, akin to human decision-making, is a challenging task, especially in situations characterized by diverse external environments and control objects, as opposed to a single, limited, and specific context. The decision-making process involves multiple steps, including reasoning and planning, before an action is executed, making it a particularly demanding challenge.

On the basis of these challenges, we propose an LLM-agent-based robot control system (LA-RCS) that minimizes human intervention and considers complex user requests, taking into account the surrounding environment of the dynamic object (CAROBO). To minimize human intervention and undertake appropriate actions to satisfy user requests, the proposed robot control system LA-RCS performs several repetitive functional procedures, specifically monitoring, observation, planning, and feedback processes based on dual agents to observe the external environment, feeding back its own actions until the user request is completely fulfilled and then constructing control commands appropriate for the user request. In addition, to execute commands more appropriately according to CAROBO's external environment on the basis of user request, we used the LLM agent with several shot-prompting methods. Through the system configuration, LA-RCS ultimately shows strong performance, enhanced autonomy, continuous performance, and appropriate command inference processes for the given intricate user requests based on the external environment of CAROBO.

To evaluate the performance rate of LA-RCS on the basis of the robot's (CAROBO) external information in the form of user requests and the efficiency of its autonomous ability when using two different LLM agents (GPT-4-Turbo and GPT-4o), we considered various user request situations during the testing process. These situations are constructed from four cases, and evaluations are conducted with five queries per case with different LLM agents. The result of the evaluation with various testing situations involved quantitative figures, emphasizing the versatility, autonomy, and expandability of robot control systems in terms of their ability to perform intricate user requests successfully.

The proposed LA-RCS is a sophisticated automation system capable of producing optimized results and considering the robot's external circumstances based on intricate user requests and considering the robot's circumstances while also minimizing human intervention during the control decision process. The designed system is also appropriate for versatile situations that require automation and can be used in many applications in other research areas. We anticipate that this study can serve as a valuable resource for the robotics research community and can stimulate further advancements in this field.

2. Related Work

2.1 LLM agents

LLM-based autonomous agents have demonstrated remarkable capabilities in reasoning, utilizing tools, and adapting to new observations across various real-world tasks.(30–32) These agents enable LLMs to perform more complex operations compared with traditional LLMs by leveraging human-like decision-making systems. AutoGPT,(33) a pioneering agent, processes user requests by decomposing the actions of LLMs into thoughts, reasoning, and critiques. TaskWeaver(34) is a code-centric agent framework that converts user requests into executable subtasks through the Python code. LangChain,(35) another LLM agent framework, assists LLMs in utilizing a variety of custom tools, such as retrieval-augmented generation and chain-of-thought reasoning. Utilizing these tools, agents can solve a wide range of problems, including the manipulation of objects, control of GUIs, and the calling of APIs. Multi-agent LLMs, which facilitate multi-agent conversations, have emerged as frameworks capable of solving more powerful and complex problems. This architecture effectively assigns tasks to individual agents with specific strengths and enables collaboration or competition among agents to handle complex tasks efficiently. AutoGen(36) is a framework that customizes each agent and utilizes conversations between agents to leverage their specific strengths, thereby enabling a multi-agent system to address user requests effectively. Similarly, frameworks such as CrewAI,(37) MetaGPT,(38) LangGraph,(39) and AutoAgent(40) build systems in which multiple agents, each with unique roles, can seamlessly perform complex tasks. UFO(18) employs a dual-agent framework to meticulously observe and analyze the GUI and to control the information of Windows applications, further highlighting the diverse applications of multi-agent LLM frameworks in real-world scenarios.

2.2 LLM agents for robot task planning

The application of multimodal LLMs for robot task planning has been gaining significant traction recently.(41–44) Previous studies often utilized few-shot or zero-shot methods to establish task plans for robots.(45) Various techniques have emerged when employing LLMs in robot task planning.

For instance, Text2Motion(46) and Do As I Can, Not As I Say(43) have demonstrated the use of value functions to enable LLMs to perform task planning for robots. Approaches such as RT-2, ChatGPT for Robotics,(47) and ProgPrompt(48) have shown that robot task planning can be successfully achieved through few-shot prompting and chain-of-thought reasoning,(49) and with the ReAct framework.

Multi-agent LLM systems have also been applied to robot task planning, showing substantial performance improvements. SMART-LLM, for example, converts high-level task instructions into multi-robot task plans, allowing multiple robots to execute complex tasks collaboratively. Each step in SMART-LLM(50) utilizes LLM prompts to decompose tasks, form teams, and allocate tasks for execution efficiently. However, existing approaches have primarily focused on task decomposition and interactions among tasks within the task planning domain, without addressing the interpretation of high-level commands, situational analysis, and without controlling robots through feedback mechanisms.

Our system utilizes a dual-agent framework inspired by UFO(18) to decompose high-level task commands for robot interfaces. This framework emphasizes the hierarchy of commands performed by each agent, enhancing their ability to interpret and execute commands through communication. The proposed method enables task planning to adapt in real time to changing environments, allowing robots to analyze the environment while remaining focused on user commands.

3. Architecture of LA-RCS

LA-RCS is an innovative multimodal agent designed for interaction between CAROBO and the external environment. LA-RCS interprets user requests expressed in natural language and decomposes the requests into step-by-step subtasks. The agent observes the external environment and manipulates CAROBO to achieve the overall goal. Through these capabilities, it can effectively accomplish complex tasks. Section 3.1 provides a general overview of the LA-RCS design, followed by a detailed discussion of each core component.

3.1 LA-RCS

Figure 1 illustrates the overall structure of LA-RCS, which operates on a dual-agent framework consisting of the following two key components:

- **Host Agent**: This agent is responsible for constructing a Global Plan to execute the user request.
- **App Agent**: This agent receives the Global Plan from the Host Agent and iteratively performs tasks to fulfill the user request on the basis of the plan.

Both agents utilize a large multimodal model to comprehend observations and execute user requests. They control CAROBO through self-determined actions using Control Functions (commands).

Figure 2 shows the workflow algorithm for the LA-RCS system. Upon receiving a User Request, the Host Agent initially analyzes the request. CAROBO provides the Host Agent with observations and sensor data to facilitate the Host Agent's understanding of its current state. On the basis of this information, the Host Agent constructs a Global Plan, which is then transmitted to the App Agent.

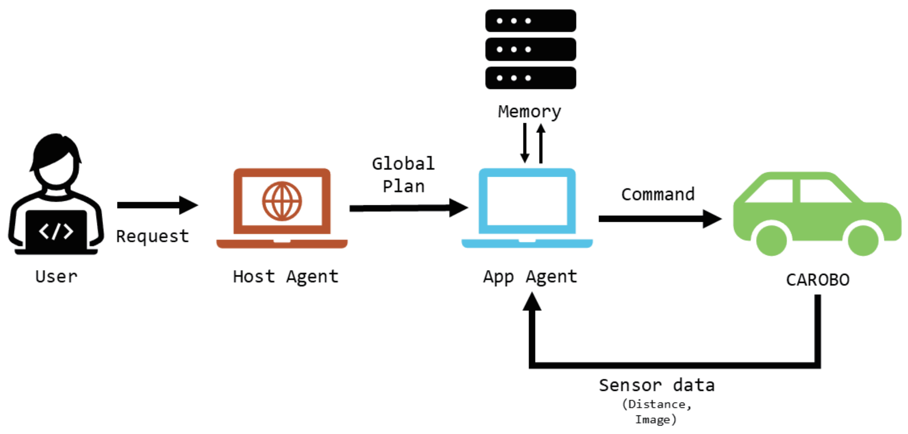


Fig. 1. (Color online) Overall architecture of LA-RCS.

```
Algorithm 1: Workflow Algorithm for the LA-RCS System
Input: User Request $U$
$S \leftarrow$ CAROBO.get_sensor_data()
Global Plan $\leftarrow$ HostAgent($U$, $S$)
do
    $M \leftarrow$ Memory.get_memory()
    $S \leftarrow$ CAROBO. get_sensor_data()
    Command $\leftarrow$ AppAgent($M$, $S$, Global Plan)
    Action $\leftarrow$ CAROBO(Command)
    Memory.update_memory(Command)
while Command["STATUS"] $\neq$ "FINISH"
end
```

Fig. 2. LA-RCS workflow algorithm.

Once the Global Plan is established, the App Agent begins to execute it. The App Agent's decision-making process is informed not only by the received Global Plan but also by the observations and sensor data. After each command is executed, the App Agent receives feedback from the external environment, allowing it to adjust subsequent commands as needed. The App Agent selects appropriate Control Functions (Commands) to manipulate CAROBO and transmits these to the system. This process, carried out by the App Agent, is repeated until it is determined that the User Request has been successfully completed.

3.2 Host Agent

The Host Agent constructs a comprehensive global plan to orchestrate the entire task, utilizing memory, which includes past task records, information about the external environment, and data collected by CAROBO from its surroundings. The Host Agent takes the following information as input:

- **User Request**: The robotic action requested by the user.
- **Sensor Data**: Data from CAROBO's sensors (camera, ultrasonic sensors).

Through Vision Data and Sensor Data, the Host Agent comprehends the current state and can constrain the selection of Control Functions for robot movement. This diverse input framework enhances the Host Agent's capability to fulfill user requests.

Utilizing all input information, the Host Agent employs an LLM to generate the following outputs:

- **Global Plan**: Action plan to fulfill the user's request.
- **Observation**: Detailed description of Vision Data.
- **Thoughts**: Logical next steps required to complete the given task.
- **Comment**: Progress status and information to be provided.

Prompting the Host Agent to provide its observations and thoughts encourages it to analyze the status, enhancing logical consistency and interpretability while assessing the task progress. During this process, the Host Agent generates outputs designed to ensure a clear analysis of the current situation when constructing the Global Plan, provides the underlying process and logic behind the decisions, and explains the progress to the user or responds to their inquiries.

3.3 App Agent

The App Agent functions as a downstream entity following the Host Agent for executing CAROBO's actions to fulfill the user requests. The App Agent accepts the following as inputs:

- **User Request**: The robotic action requested by the user.
- **Global Plan**: Action plan to fulfill the user's request.
- **Memory**: Previous thoughts, comments, actions, and execution results.
- **Sensor Data**: Data used to determine action initiation or termination based on the interaction between CAROBO and objects (camera, ultrasonic sensors).

As previously explained, the App Agent determines CAROBO's actions on the basis of the Global Plan to fulfill the User Request. Memory provides CAROBO's past actions, allowing the App Agent to analyze and reduce the probability of repeating inefficient or meaningless actions. On the basis of these inputs, the App Agent analyzes the information and produces the following outputs:

- **Comment**: Progress status and information to be provided.
- **Control Function (Command)**: Action to be performed by CAROBO.
- **Observation**: Analyzed Vision Data.
- **Status**: Task status, "**CONTINUE**" if additional action is needed and "**FINISH**" if the action is completed.

The App Agent determines the next step on the basis of these output states. It repeatedly undertakes observations through Vision Data and CAROBO control until the action is completed, i.e., until the Status reaches the **FINISH** state. These outputs are continuously stored in Memory, contributing to the App Agent's correct decision-making outcome in fulfilling user requests.

3.4 CAROBO

We made use of CAROBO as a robot platform to implement LA-RCS. CAROBO is a car-type robot built using the Raspbot from Yahboom. To provide Vision Data to LA-RCS, CAROBO includes a camera. For Sensor Data, it includes ultrasonic sensors and infrared obstacle detection sensors. Additionally, we integrated motors for robot actions, a servo motor for camera adjustments, and a buzzer to enable sound functionality. The communication between CAROBO and LA-RCS was configured to utilize socket communication.

The control functions received by the robot are predefined in CAROBO. The following functions exist in CAROBO:

- **Car forward**: Command for the robot to move forward. It has a feature of turning right whenever it is too close to an object in front.
- **Car back**: Command for the robot to move backward.
- **Car left**: Command for the robot to turn left.
- **Car right**: Command for the robot to turn right.
- **Camera move**: Command to adjust the angle of the robot's camera.
- **Buzzer**: Command to activate the robot's buzzer.

The App Agent assesses the situation and selects one of these Control Functions (Commands) to execute robot control. After executing a command, CAROBO saves the following output to the memory:

- **Action**: The robot performs the action according to the command.
- **Vision Data**: Vision data for confirming the robot's next action or action termination.
- **Sensor Data**: Sensor data for confirming action or action termination based on the distance between the robot and the objects.

4. Evaluation

In this section, we evaluate LA-RCS's performance in completing user requests. The evaluation is conducted through a quantitative analysis and case studies.

4.1 Benchmark and metrics

To assess the performance of LA-RCS comprehensively, we developed a benchmark consisting of 20 user requests. These requests were designed across four domains: object detection, command execution, obstacle navigation, and situational awareness. This approach ensures diversity and comprehensiveness in the evaluation. Five user requests were designed for each domain, resulting in a total of 20 requests.

Given that no pre-existing agent effectively replicates the role of the proposed LA-RCS, we selected the GPT-4-Turbo and GPT-4 models as control baselines for comparison. Unlike LA-RCS, these models lack the inherent capability to interact directly with CAROBO, a limitation that impacts their operation dynamics in completing user requests. Therefore, to facilitate this interaction, we directed the models to provide step-by-step guidance, enabling them to simulate the completion of tasks.

A human operator subsequently executed the steps outlined by these models, acting as a surrogate for the direct control that LA-RCS provides autonomously. This approach allowed us to assess the practical effectiveness of these baseline models in terms of task fulfillment and user interaction when contrasted with the autonomous functionalities embedded within LA-RCS.

We evaluated each User Request from two perspectives: success and step count. Success indicates that LA-RCS successfully completed the User Request. Failure is determined when LA-RCS incorrectly judges a request as completed when it has not been, or when it exhibits repetitive, meaningless movements for a certain number of steps. A step represents an action taken by LA-RCS to complete the request.

Given the potential instability of GPT-4o, which may generate different outputs each time, we conducted three tests for each request and selected the one with the highest completion rate. This approach is consistent for the other baselines as well.

4.2 Performance evaluation

We conducted a quantitative comparison of LA-RCS using our proprietary dataset, as illustrated in Table 1. When using GPT-4o as the backend LLM for LA-RCS, the success rate was 90%, twice that of GPT-4o (Human Surrogate), which showed a success rate of 45%. This result demonstrates the considerable effectiveness of LA-RCS in the domain of robot planning. Additionally, the fewer the steps required for command execution, the significantly greater the accuracy and speedthat LA-RCS can perform tasks.

Furthermore, as shown in Fig. 3, the capabilities across each domain improved when the LA-RCS framework was applied. The shortcomings of the Human Surrogate (GPT) compared with those of LA-RCS can be attributed to its inability to interact with the external environment directly. This limitation reduces its accuracy owing to a lack of adaptability to changes in the environment and responsive adjustments.

Tables 2–5 present detailed success and failure rates for user requests in the four domains, namely, object detection, command execution, obstacle navigation, and situational awareness, respectively.

User requests for the object detection domain are presented in Table 2. In this domain, the Agent composed of GPT-4-Turbo successfully completed four out of five tasks, demonstrating an 80% success rate, whereas the Agent composed of GPT-4o achieved a 100% success rate, that is, completing all five tasks.

Table 1
Performance comparison achieved by LA-RCS.

Model	Average step count	Success rate (%)
LA-RCS (GPT-4 Turbo)	7.9	60
LA-RCS (GPT-4o)	6.9	90
GPT-4o (Human Surrogate)	11.2	30
GPT-4 Turbo (Human Surrogate)	9.5	45

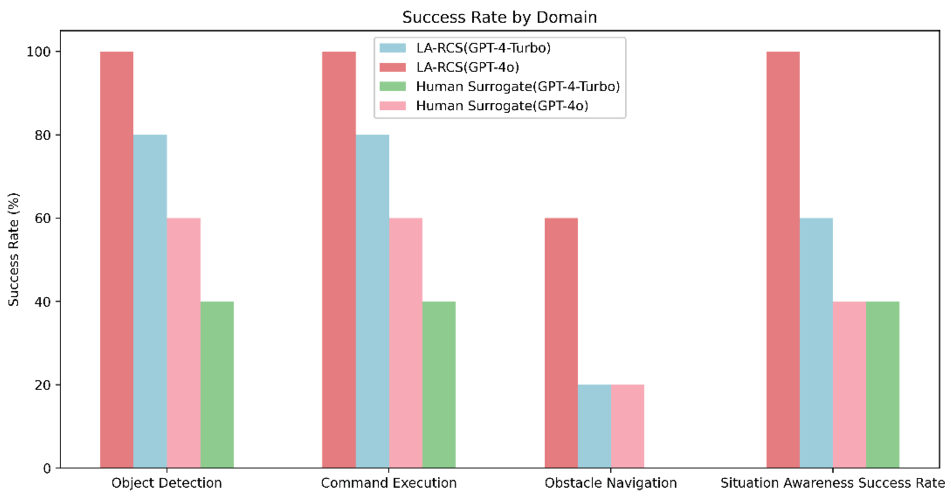


Fig. 3. (Color online) Success rate by domain.

Table 2
Object detection results.

Request	GPT-4-Turbo		GPT-4o	
Move around and search for the location of the refrigerator	Success	6	Success	9
Look for any people in the immediate vicinity	Success	8	Success	4
Move around and search for any people in the surrounding area	Failure	20	Success	11
If a rectangular object is seen, move towards it and activate the buzzer	Success	4	Success	2
Find yellow obstacles and tell me what is written on them	Success	0	Success	0

Table 3
Command execution results.

Request	GPT-4-Turbo		GPT-4o	
Rotate in a 0.6 m^2 shape	Failure	5	Success	9
Lift your head, identify a human face, and describe it	Success	3	Success	3
Rotate twice on the spot	Success	1	Success	8
Move into a zigzag pattern at a 30 deg angle for 2 m	Success	5	Success	10
Move backwards 0.4 m and sound the buzzer, repeat this 5 times	Success	12	Success	12

Table 4
Obstacle navigation results.

Request	GPT-4-Turbo		GPT-4o	
Move forward avoiding obstacles	Failure	16	Success	7
Move forward for a total of 2 m, turning right and activating the buzzer when obstacles appear	Success	8	Success	5
Move to find a Bosch box while avoiding obstacles	Failure	15	Success	8
Rotate once while observing the surroundings, then move 1 m in a direction without obstacles	Failure	5	Failure	15
After observing obstacles in front, move behind the observed object, stop, and activate the buzzer	Failure	3	Failure	5

Table 5
Situation awareness results.

Request	GPT-4-Turbo		GPT-4o	
Describe the features of the object in front.	Success	0	Success	0
Detect the surroundings and describe only navy-colored objects in Korean	Success	6	Success	7
Move forward 0.5 m, observe the surroundings, and tell me the name of the box	Success	3	Success	7
After moving 2 m, if there is a piece of paper in front, print out what is written on it	Failure	3	Success	10
From the paper observed in front, tell me the contact information of the Ministry of Science, ICT, and Future Planning	Failure	4	Success	0

User requests for the command execution domain are presented in Table 3. In this domain, the Agent composed of GPT-4-Turbo successfully completed four out of five tasks, demonstrating an 80% success rate, while the Agent composed of GPT-4o achieved a 100% success rate, that is, completing all five tasks.

User requests for the obstacle navigation domain are presented in Table 4. In this domain, the Agent composed of GPT-4-Turbo successfully completed one out of five tasks, demonstrating a 20% success rate, while the Agent composed of GPT-4o successfully completed four out of five tasks, demonstrating a 60% success rate.

The primary reason for failure was the insufficient data to accurately assess the external environment on the basis of the camera and sensor input. Owing to this limitation, the agent exhibited repetitive behavior without meaningful progress, ultimately exceeding the allowed step count and terminating the task prematurely or failing to complete the external exploration. This indicates that the current system has difficulty in adjusting its behavior when real-time environmental data is incomplete or ambiguous, indicating the necessity for enhanced sensor fusion and environmental perception capabilities.

User requests for the situation awareness domain are presented in Table 5. In this domain, the Agent composed of GPT-4-Turbo successfully completed three out of five tasks, demonstrating a 60% success rate, while the Agent composed of GPT-4o achieved a 100% success rate by completing all five tasks.

In conclusion, the overall evaluation results indicate that the proposed LA-RCS system, when utilizing GPT-4o, exhibited significantly enhanced performance compared with GPT-4 Turbo and human-surrogate-based approaches. GPT-4o demonstrated superior generalization capabilities, enabling more precise reasoning and decision-making in response to complex user requests. Specifically, LA-RCS with GPT-4o achieved a 90% success rate, surpassing the performance of GPT-4 Turbo and human surrogate models, which demonstrated lower success rates and required a greater step count to complete tasks.

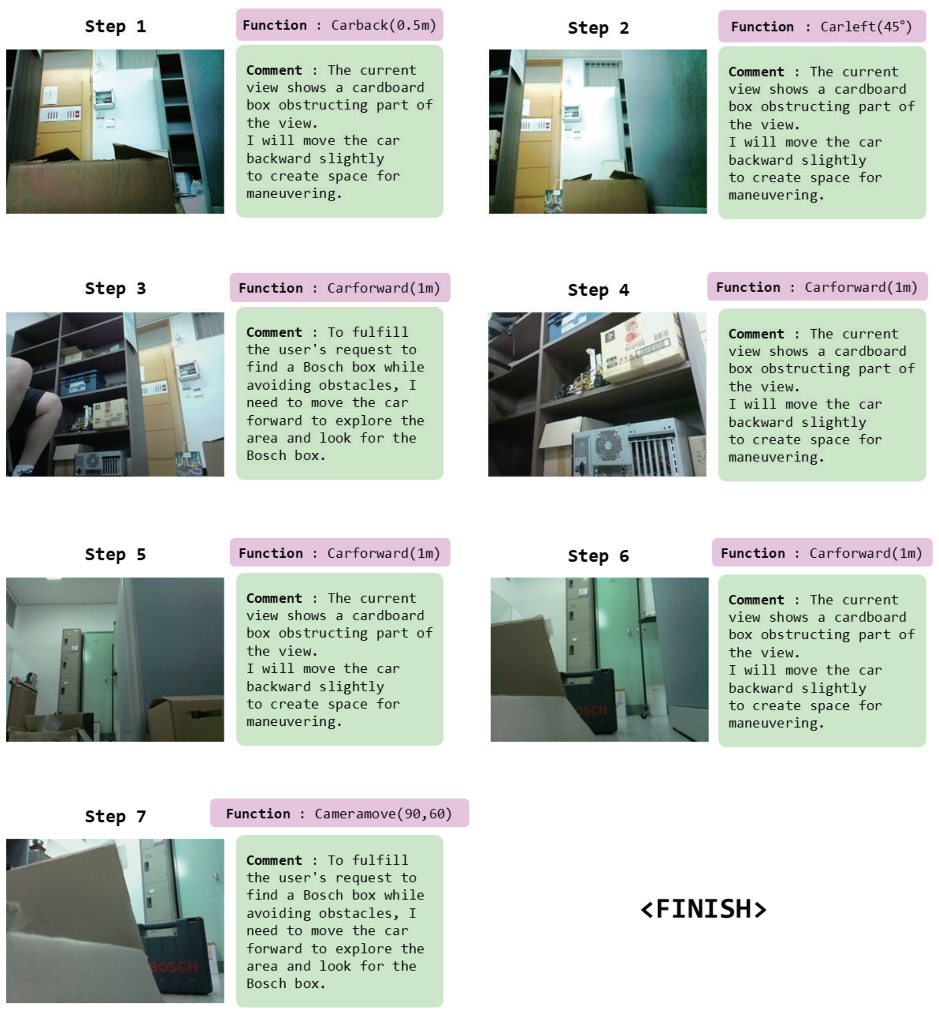
Nevertheless, failure cases were primarily observed when LA-RCS with GPT-4o encountered constraints in receiving external environmental data or limitations in the camera's field of view. Under such circumstances, the system was unable to accurately assess the surrounding environment, resulting in repetitive actions and ultimately leading to termination because the allowed step count was exceeded. These findings suggest that enhancing sensor fusion and improving real-time environmental perception would be beneficial in further increasing the system's robustness and adaptability in dynamic environments.

4.3 Case study

4.3.1 Exploration and object detection in the surrounding environment

In Fig. 4, we tasked LA-RCS with the following request: "Find the box that says Bosch." This scenario involves placing various objects in the surrounding environment. For LA-RCS to execute a request, it must continuously assess the current space, constantly search for the given objects, and upon discovering the specified target, complete the challenging request.

To navigate such a situation, LA-RCS autonomously determines the need to execute the Car back function to secure its view. On the basis of the acquired visual information, it analyzes where to move next and proceeds forward. Subsequently, when identifying an object that is



Step 1

Function : Carback(0.5m)

Comment : The current view shows a cardboard box obstructing part of the view.
I will move the car backward slightly to create space for maneuvering.

Step 2

Function : Carleft(45°)

Comment : The current view shows a cardboard box obstructing part of the view.
I will move the car backward slightly to create space for maneuvering.

Step 3

Function : Carforward(1m)

Comment : To fulfill the user's request to find a Bosch box while avoiding obstacles, I need to move the car forward to explore the area and look for the Bosch box.

Step 4

Function : Carforward(1m)

Comment : The current view shows a cardboard box obstructing part of the view.
I will move the car backward slightly to create space for maneuvering.

Step 5

Function : Carforward(1m)

Comment : The current view shows a cardboard box obstructing part of the view.
I will move the car backward slightly to create space for maneuvering.

Step 6

Function : Carforward(1m)

Comment : The current view shows a cardboard box obstructing part of the view.
I will move the car backward slightly to create space for maneuvering.

Step 7

Function : Cameramove(90,60)

Comment : To fulfill the user's request to find a Bosch box while avoiding obstacles, I need to move the car forward to explore the area and look for the Bosch box.

<FINISH>

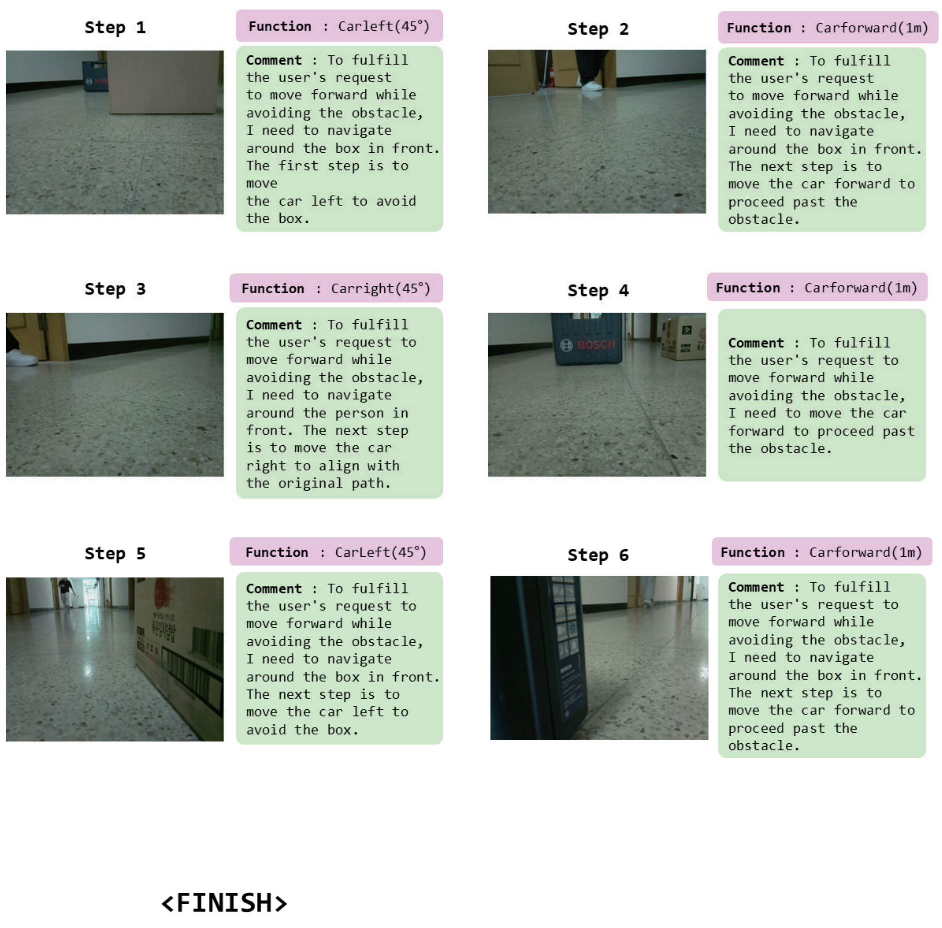
Fig. 4. (Color online) Detailed example of completing user request: "Find the box that says Bosch".

predicted to be the “Bosch Box”, it raises its head to confirm the match and completes the request.

4.3.2 Object passage

In Fig. 5, we tasked LA-RCS with the following request: “Move forward avoiding obstacles.” This scenario involves navigating around various obstacles. To execute a request, LA-RCS must continuously assess the current space and determine how to move autonomously, making it a challenging task. To handle such a situation, LA-RCS observes the external environment and calculates its actions to move forward.

These two outcomes demonstrate that LA-RCS can analyze how to fulfill the user’s request, plan accordingly, move CAROBO, and continuously control its movements while observing the external environment.



Step 1

Function : Carleft(45°)

Comment : To fulfill the user's request to move forward while avoiding the obstacle, I need to navigate around the box in front. The first step is to move the car left to avoid the box.

Step 2

Function : Carforward(1m)

Comment : To fulfill the user's request to move forward while avoiding the obstacle, I need to navigate around the box in front. The next step is to move the car forward to proceed past the obstacle.

Step 3

Function : Carright(45°)

Comment : To fulfill the user's request to move forward while avoiding the obstacle, I need to navigate around the person in front. The next step is to move the car right to align with the original path.

Step 4

Function : Carforward(1m)

Comment : To fulfill the user's request to move forward while avoiding the obstacle, I need to move the car forward to proceed past the obstacle.

Step 5

Function : CarLeft(45°)

Comment : To fulfill the user's request to move forward while avoiding the obstacle, I need to navigate around the box in front. The next step is to move the car left to avoid the box.

Step 6

Function : Carforward(1m)

Comment : To fulfill the user's request to move forward while avoiding the obstacle, I need to navigate around the box in front. The next step is to move the car forward to proceed past the obstacle.

<FINISH>

Fig. 5. (Color online) Detailed example of completing user request: “Move forward avoiding obstacles”.

5. Conclusion and Future Work

In our research, we explored the LLM Agent system, where CAROBO autonomously performs given tasks through observation and feedback. The LA-RCS system is based on a dual-agent framework, separating a macro-level task planning agent and an agent that physically executes the tasks, engaging in conversation to accomplish the given tasks. This approach enables the agent to resolve unfamiliar situations independently, decompose complex tasks into manageable parts, and provide feedback to handle unexpected anomalies. The refinement of this method suggests the potential for developing a system capable of executing tasks with minimal human intervention in various scenarios.

However, our study revealed certain limitations that need to be addressed to improve system performance. First, the system exhibited delays in response time, particularly when handling dynamic environments that require real-time adjustments. The delays in generating and executing commands affected the system's overall responsiveness and efficiency. Second, high computational costs associated with complex decision-making processes and repeated prompt inferences imposed a burden on system efficiency, limiting real-time performance and scalability. Additionally, the limited command execution capabilities of the robot restricted the completion of certain tasks. Addressing these issues by optimizing the decision-making pipeline, improving communication efficiency between agents, and reducing computational overhead can further enhance system performance in time-sensitive and computationally demanding scenarios.

In our future work, we plan to enhance the system's responsiveness and adaptability by improving the efficiency of the decision-making process and expanding the range of supported commands. Additionally, we will evaluate the efficacy of the proposed system in more sophisticated environments beyond the current setup, which primarily involves vision camera position control and vehicle speed control strategies.

Acknowledgments

This work was supported by a grant from the Korea Institute for Advancement of Technology (KIAT) funded by the Korean Government (MOTIE) [P0024163, Development of Regional Innovation Cluster (RIC)]. This work also was supported by the Korea Research Institute for Defense Technology Planning and Advancement (KRIT) funded by the Defense Acquisition Program Administration (DAPA) (KRIT-CT-23-021).

References

1 J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus: arXiv preprint (2022). arXiv:2206.07682
2 H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian: arXiv preprint (2023). arXiv:2307.06435
3 S. Hamza Cherif, L. F. K. Tani, and N. Settouti: Adv. Technol. Innovation **9** (2024) 129. https://doi.org/10.46604/aiti.2024.13523
4 A. Pimpalkar and J. R. Raj: Adv. Technol. Innovation **8** (2023) 254. https://doi.org/10.46604/aiti.2023.11743

5 T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa: arXiv preprint (2022). arXiv:2205.11916
6 L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe: arXiv preprint (2022). arXiv:2203.02155
7 T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei: arXiv preprint (2020). arXiv:2005.14165
8 OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, and 181 additional authors not shown: arXiv preprint (2023). arXiv:2303.08774
9 H. Sun, Y. Zhuang, L. Kong, B. Dai, and C. Zhang: arXiv preprint (2023). arXiv:2305.16653
10 X. Chen, X. Song, L. Jing, S. Li, L. Hu, and L. Nie: arXiv preprint (2022). arXiv:2207.07934
11 J. Yang, H. Zhang, F. Li, X. Zou, C. Li, and J. Gao: arXiv preprint (2023). arXiv:2310.11441
12 J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, R. Ring, E. Rutherford, S. Cabi, T. Han, Z. Gong, S. Samangooei, M. Monteiro, J. Menick, S. Borgeaud, A. Brock, A. Nematzadeh, S. Sharifzadeh, M. Binkowski, R. Barreira, O. Vinyals, A. Zisserman, and K. Simonyan: Flamingo: arXiv preprint (2022). arXiv:2204.14198
13 Y.-C. Chen, L. Li, L. Yu, A. El Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu: arXiv preprint (2019). arXiv:1909.11740
14 Z. Yang, L. Li, K. Lin, J. Wang, C.-C. Lin, Z. Liu, and L. Wang: arXiv preprint (2023). arXiv:2309.17421
15 X. Zhang, Y. Lu, W. Wang, A. Yan, J. Yan, L. Qin, H. Wang, X. Yan, W. Y. Wang, and L. R. Petzold: arXiv preprint (2023). arXiv:2311.01361
16 B. Zheng, B. Gou, J. Kil, H. Sun, and Y. Su: arXiv preprint (2024). arXiv:2401.01614
17 C. Zhang, L. Li, S. He, X. Zhang, B. Qiao, S. Qin, M. Ma, Y. Kang, Q. Lin, S. Rajmohan, D. Zhang, and Q. Zhang: arXiv preprint (2024). arXiv:2402.07939
18 T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom: arXiv preprint (2023). arXiv:2302.04761
19 J. Zhang, K. Wang, R. Xu, G. Zhou, Y. Hong, X. Fang, Q. Wu, Z. Zhang, and H. Wang: arXiv preprint (2024). arXiv:2412.06224
20 K. Mei, Z. Li, S. Xu, R. Ye, Y. Ge, and Y. Zhang: arXiv preprint (2024). arXiv:2403.16971
21 N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao: arXiv preprint (2023). arXiv:2303.11366
22 X. Deng, Y. Gu, B. Zheng, S. Chen, S. Stevens, B. Wang, H. Sun, and Y. Su: arXiv preprint (2023). arXiv:2306.06070
23 T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest, and X. Zhang: arXiv preprint (2024). arXiv:2402.01680
24 J. Zhang, K. Wang, R. Xu, G. Zhou, Y. Hong, X. Fang, Q. Wu, Z. Zhang, and H. Wang: arXiv preprint (2024). arXiv:2402.15852
25 S. Patil, V. Vasu, and K. V. S. Srinadh: Discover Mech. Eng. **2** (2023) 13. https://doi.org/10.1007/s44245-023-00021-8
26 N. Wake, A. Kanehira, K. Sasabuchi, J. Takamatsu, and K. Ikeuchi: IEEE Rob. Autom. Lett. **9** (2024) 10567. https://doi.org/10.48550/arXiv.2311.12015
27 J. Wang, Z. Wu, Y. Li, H. Jiang, P. Shu, E. Shi, H. Hu, C. Ma, Y. Liu, X. Wang, Y. Yao, X. Liu, H. Zhao, Z. Liu, H. Dai, L. Zhao, B. Ge, X. Li, T. Liu, and S. Zhang: arXiv preprint (2024). arXiv:2401.04334
28 H. Liu, Y. Zhu, K. Kato, A. Tsukahara, I. Kondo, T. Aoyama, and Y. Hasegawa: IEEE Rob. Autom. Lett. **9** (2024) 6904. https://doi.org/10.1109/LRA.2024.3415931
29 H. Liu, Y. Zhu, K. Kato, I. Kondo, T. Aoyama, and Y. Hasegawa: arXiv preprint (2023). arXiv:2308.14972

30 S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao: 2023 Int. Conf. Learning Representations (ICLR,2023).

31 Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, R. Zheng, X. Fan, X. Wang, L. Xiong, Y. Zhou, W. Wang, C. Jiang, Y. Zou, X. Liu, Z. Yin, S. Dou, R. Weng, W. Qin, Y. Zheng, X. Qiu, X. Huang, Q. Zhang, and T. Gui: Sci. China Inf. Sci. **68** (2025) 121101. https://doi.org/10.1007/s11432-024-4222-0

32 L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. X. Zhao, Z. Wei, and J. Wen: Front. Comput. Sci. **18** (2024) 186345. https://doi.org/10.1007/s11704-024-40231-1

33 AutoGPT: https://github.com/Significant-Gravitas/AutoGPT (accessed April 2024).

34 B. Qiao, L. Li, X. Zhang, S. He, Y. Kang, C. Zhang, F. Yang, H. Dong, J. Zhang, L. Wang, M. Ma, P. Zhao, S. Qin, X. Qin, C. Du, Y. Xu, Q. Lin, S. Rajmohan, and D. Zhang: arXiv preprint (2023). arXiv:2311.17541.

35 LangChain: https://github.com/langchain-ai/langchain (accessed April 2024).

36 Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang: arXiv preprint (2023). arXiv:2308.08155

37 crewAIInc: https://github.com/crewAIInc/crewAI (accessed April 2024).

38 S. Hong, X. Zheng, J. Chen, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, C. Wu, and J. Schmidhuber: arXiv preprint (2023). arXiv:2308.00352

39 Langchain-ai: https://github.com/langchain-ai/langgraph (accessed April 2024).

40 G. Chen, S. Dong, Y. Shu, G. Zhang, J. Sesay, B. F. Karlsson, J. Fu, and Y. Shi: arXiv preprint (2023). arXiv:2309.17288

41 C. E. Mower, Y. Wan, H. Yu, A. Grosnit, J. Gonzalez Billandon, M. Zimmer, J. Wang, X. Zhang, Y. Zhao, A. Zhai, P. Liu, D. Palenicek, D. Tateo, C. Cadena, M. Hutter, J. Peters, G. Tian, Y. Zhuang, K. Shao, X. Quan, J. Hao, J. Wang, and H. Bou Ammar: arXiv preprint (2024). arXiv:2406.19741

42 T. Kwon, N. Di Palo, and E. Johns: IEEE Rob. Autom. Lett. **9** (2024) 6728. https://doi.org/10.1109/LRA.2024.3410155

43 M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng: arXiv preprint (2022). arXiv:2204.01691

44 A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. González Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich: arXiv preprint (2023). arXiv:2307.15818

45 T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei: Language Models are Few-shot Learners. Advances in neural information processing systems (2020).

46 K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg: Auton. Rob. **47** (2023) 1345. https://doi.org/10.1007/s10514-023-10131-7

47 S. H. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor: IEEE Access **12** (2024) 55682. https://doi.org/10.48550/arXiv.2306.17582

48 I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg: Proc. 2023 IEEE Int. Conf. Robotics and Automation (ICRA, 2023) 11523–11530. https://doi.org/10.1109/ICRA48891.2023.10161317

49 J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou: Adv. Neural Inf. Process. Syst. **35** (2022) 24824.

50 S. S. Kannan, V. L. N. Venkatesh, and B. C. Min: Proc. 2024 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS, 2024) 12140–12147. https://doi.org/10.1109/IROS58592.2024.10802322

About the Authors

Taek-Hyun Park received his B.S. degree from National Korea Maritime & Ocean University, Korea, in 2025. and is currently pursuing his M.S. degree at the Graduate School of Data Science, Pusan National University, Korea. His research interests are in time series forecasting, deep learning, novel architecture, and intelligent control methods.

Young-Jun Choi received his B.S. degree from National Korea Maritime & Ocean University, Korea, in 2025. His research interests are in deep learning, novel architecture, and intelligence control methods.

Seung-Hoon Shin received his B.S. degree from National Korea Maritime & Ocean University, Korea, in 2023. Since 2023, he has been a navy officer at the Naval Intelligence Information System Group. His research interests are in autonomous mobility, intelligent control methods, and deep learning.

Chang-Eun Lee received his B.S. and M.S. degrees in electronics engineering from Hanyang University, Republic of Korea, in 1996 and 1998, respectively, and his Ph.D. degree in information and communication engineering from Chungnam National University, Republic of Korea, in 2017. From 1998 to 2000, he was a researcher at LG Industry System, Republic of Korea, where he worked on intelligent building automation systems. Since 2001, he has been with the Electronics and Telecommunications Research Institute (ETRI), Republic of Korea, where he has conducted research in the fields of intelligent robot systems and military artificial intelligence. His primary research interests are artificial intelligence, robot software frameworks, and distributed and cooperative unmanned systems.

Kwangil Lee received his B.S., M.S., and Ph.D. degrees from the Department of Computer Science at Chungnam National University in 1993, 1996, and 2001, respectively. From 2000 to 2002, he worked as a guest researcher at the National Institute of Standards and Technology (NIST) in the USA. Then, he worked as a research associate at the University of Maryland (2002–2004) and later at the University of Texas (2005) in the USA. He participated in the first of Korea's smart ship projects in 2007. Since then, he has participated in many smart and digitalization maritime research projects, such as E-navigation and MASS. His research interests are in autonomous ships, smart ships, e-navigation, common maritime data structures, cyber security and safety, situational awareness, generative AI, and mobile/wireless networks.