Sensors and Materials, Vol. 38, No. 7 (2026) 4259–4276
MYU Tokyo

S & M 4563

AI Agent Framework for Maritime Surveillance Using Electro-optical Targeting Systems

Ki-Woon Yeun[1,2] and Yun-Soo Choi[1*]

[1]Department of Geoinformatics, University of Seoul, Seoul 02504, Korea
[2]Hanwha Systems, Pine Avenue Bldg Tower A 7F 100, Eulji ro, Jung gu, Seoul 04551, Korea

(Received November 27, 2025; accepted July 6, 2026)

Keywords: maritime surveillance, electro-optical targeting system, EO/IR, large language model, AI agent

The Electro-optical Targeting System (EOTS) performs detection, tracking, and identification of maritime and aerial targets using EO/IR sensors. However, in maritime surveillance environments, operators must manually configure workflows such as multisensor control and detection or tracking parameter settings, which requires considerable time and manpower and hinders rapid response. To overcome these limitations, in this study, we propose an EOTS-AI Agent framework based on a large language model (LLM). The proposed framework interprets and analyzes natural language queries from maritime surveillance operators, autonomously selects appropriate tools, and executes tasks step by step while providing an intuitive and efficient natural language interface. The results demonstrate that the proposed framework reduces operator workload and enables rapid, multicondition detection and tracking, thereby significantly improving the efficiency and reliability of maritime surveillance operations.

1. Introduction

With the recent advances in generative AI (Generative AI), particularly large language models (LLMs), capabilities in natural language understanding, complex reasoning, and code generation have improved considerably, and the automation of complex business processes is rapidly progressing. In the geospatial domain as well, there has been active discussion of the so-called "geographic information system (GIS) copilot"/"autonomous GIS agent" approach, which converts user requirements expressed in natural language into executable geoprocessing workflows, and case studies have been reported in which an LLM sequentially organizes the entire pipeline—from data collection, preprocessing, and analysis to visualization—into stages of understanding, planning, and tool execution. For example, GeoGPT proposes and validates a framework that structures natural language input into a sequence of procedures—"think → plan → sequential execution of GIS tools (execute)"—to produce the final analytical result, thereby suggesting a direction that meets the practical requirement of orchestrating the execution of multiple tools, which is inherently necessary in geospatial tasks.(1)

*Corresponding author: e-mail: choiys@uos.ac.kr
https://doi.org/10.18494/SAM6192

ISSN 0914-4935
https://myukk.org/

Meanwhile, from the perspective of geospatial data retrieval, geospatial data question answering systems that combine knowledge graphs with LLMs have been proposed, demonstrating that even nonexpert users can perform data exploration that includes complex spatiotemporal constraints through a natural-language-based interface. In these studies, large-scale geospatial datasets and metadata are normalized into a domain ontology, and the LLM infers relevant nodes and relationships while taking spatiotemporal conditions and semantic constraints into account, thereby providing greater expressiveness and flexibility than traditional keyword-based search.[(2)]

To reliably connect natural language instructions all the way to actual tool invocation, execution, and verification, standardized tool interfaces and systematic tool discovery mechanisms are essential, beyond *ad hoc* integrations at the level of individual applications. The Model Context Protocol (MCP) proposed by Anthropic is an open standard that standardizes secure bidirectional connections between AI applications and various data sources and business tools; through a simple architecture in which developers expose data via MCP servers and MCP clients connect to those servers, it unifies fragmented individual connectors under a single protocol. In this way, heterogeneous systems such as in-house content repositories, business systems, and development tools are integrated under one specification, providing a foundation for efficiently accessing necessary information while maintaining contextual consistency. Practical means of adoption are provided alongside the MCP specification, including software development kits (SDKs), clients that support local MCP servers, and public MCP server registries, and as various commercial and open-source developer tool ecosystems participate, MCP's status as a "standard interface for AI that connects data sources, tools, and workflows" is being reinforced. Furthermore, the MCP Registry systematically supports tool discoverability by providing clients with a list of servers.[(3)]

While MCP thus presents a standard interface for tool integration, in large-scale tool ecosystems, there have been reports of limitations in which the LLMs' tool selection performance is easily degraded by prompt bloat and complexity of choice. Retrieval-augmented Generation for MCP (RAG-MCP) proposes a structure in which, instead of indiscriminately injecting all tool schemas registered in MCP into the prompt, an external vector index is used to select only a small number of tools that are semantically highly related to the query in the order of search → validation → invocation, and only the schemas of those tools are included in the prompt. This design reduces unnecessary schema injection, lowering token costs while also mitigating selection errors and hallucinations that intensify as the number of tools grows. In various stress tests assuming an MCP environment, RAG-MCP showed more stable tool selection performance and response consistency than simple injection approaches, and it was confirmed that the system can be scaled in situations where tools are continuously added simply by updating the index, without retraining. Ultimately, the RAG-MCP approach, which "separates tool discovery into a search process", provides a practical design principle for simultaneously improving selection accuracy and cost efficiency in large MCP-based tool sets.[(4)]

In this study, we extend the discussion of LLM-based tool orchestration to the context of maritime surveillance operations. The Electro-optical Targeting System (EOTS) is a key device that performs target detection, tracking, and identification using EO and IR sensors. However, in

actual operation, there is a limitation in that the operator must manually configure multistep procedures such as multisensor control, mode switching, parameter setting and adjustment, and exception handling. In particular, in maritime surveillance missions, operating procedures branch in complex ways depending on target type, weather and visibility conditions, and rules of engagement, which leads to high dependence on highly skilled operators and makes it difficult to consistently apply optimal sensor operation strategies in real time.

To address this, we propose, in this paper, an EOTS-AI Agent framework that directly integrates an LLM into the EOTS operating system. The proposed system takes natural language queries as input, performs MCP-based tool discovery, and is designed to safely construct and execute EOTS control commands through consistency verification of the selected tool schemas, execution monitoring, and an automatic correction loop. In particular, considering that EOTS is operated in a closed network environment isolated from external networks for the surveillance of secure areas, the framework is implemented using an open-source LLM without relying on commercial cloud application programming interfaces (APIs).

In this framework, a Qwen-series model, designed to support more than 100 languages, is adopted as the core LLM. Qwen possesses language understanding performance that is evaluated as comparable to or exceeding that of state-of-the-art models, especially for Asian languages including Chinese, and Korean is also included among its primary supported languages. This provides a basis for performing consistent question answering and reasoning without loss of meaning even in environments where Korean-based operational commands and other multilingual metadata coexist. In addition, by designing an interface that complies with the EOTS integration protocol, the multilingual LLM capabilities are safely linked to actual sensor control commands, thereby ensuring both the traceability and reliability of operating procedures.

The remainder of this paper is organized as follows. In Sect. 2, we review related work, and in Sect. 3, we describe in detail the architecture of the proposed EOTS-AI Agent system. In Sect. 4, we present the scenario-based evaluation methodology, performance test results, and limitations, and in Sect. 5, we provide conclusions and directions for future research.

2. Related Work

2.1 LLM

Recently, LLMs, exemplified by OpenAI's GPT series, have been reported to consistently outperform previous-generation models across a wide range of tasks such as natural language understanding, reasoning, code generation, summarization, and translation, as a result of advances in pretraining using large-scale web, textual, code, and multimodal data, as well as in alignment techniques. In particular, as their ability to decompose and execute complex instructions step by step and to generate coherent responses within long contexts has improved significantly, their scope of use has expanded from ordinary conversational agents to general-purpose platforms that support complex interactions such as domain-specific tool invocation, code authoring and modification, and comprehensive question answering over multiple documents.[5]

Meanwhile, in terms of LLM alignment, principle-based alignment methods such as "Constitutional AI" proposed by Anthropic are emerging as an important line of research. This approach is aimed at training the model to suppress harmful or biased outputs while maintaining usefulness and honesty by having another AI critique and revise the model's responses against a set of predefined principles (a constitution) and iteratively improving the original model using that feedback. Such principle-based alignment research is evolving in conjunction with efforts to secure safe and consistent response quality while improving long-context handling, understanding of complex instructions, and transparency of the reasoning process, and more recently, it has been extended to multimodal LLMs that handle multiple input modalities, such as images in addition to text.[6]

In the line of open-weight LLM research, the latest version of the Qwen series, Qwen3, points to one possible direction. According to the Qwen3 Technical Report, Qwen3 consists of dense models and Mixture-of-Experts (MoE) models with parameter scales ranging from 0.6B to 235B. The detailed architectures of each are summarized in Table 1 (dense models) and Table 2 (MoE models). Qwen3 is designed to provide a continuous scale from lightweight to ultralarge models within a single family, and by integrating, within a single model, a thinking mode for complex multistep reasoning and a nonthinking mode for fast responses, and introducing the notion of a thinking budget that allows the amount of reasoning to be adjusted in accordance with task difficulty, it exemplifies a line of research on finely controlling the balance between reasoning performance and latency. On the pretraining side, it is characterized by having substantially enhanced multilingual and multidomain performance owing to the use of a massive corpus of about 36 trillion tokens that cover various domains such as code, mathematics, science and engineering (STEM), reasoning, and literature, and by expanding the supported languages from 29 to 119 languages and dialects. Through this design, Qwen3 achieves performance close to or competitive with commercial closed-source models on various benchmarks such as code generation, mathematical reasoning, and agent tasks, while being an open-weight model under the Apache 2.0 license, thereby enabling broad use in the research community and in industry, including on-premises and air-gapped environments.[7]

Table 1
Model architecture of Qwen3 dense models.

Models	Layers	Heads (Q/KV)	Tie embedding	Context length
Qwen3-0.6B	28	16/8	Yes	32K
Qwen3-1.7B	28	16/8	Yes	32K
Qwen3-4B	36	32/8	Yes	128K
Qwen3-8B	36	32/8	No	128K
Qwen3-14B	40	40/8	No	128K
Qwen3-32B	64	64/8	No	128K

Table 2
Model architecture of Qwen3 MoE models.

Models	Layers	Heads (Q/KV)	#Experts (total/activated)	Context length
Qwen3-30B-A3B	48	32/4	128/8	128K
Qwen3-235B-A22B	94	64/4	128/8	128K

2.2 Research cases of AI agents in the field of geospatial information

Research on agentic GIS centered on LLMs has, in recent years, rapidly developed toward the automation of the entire "data collection–analysis–visualization" pipeline on the basis of natural language. In particular, Ning *et al.* directly addressed the bottleneck of "data acquisition", a prerequisite task in spatial analysis, and proposed an autonomous GIS data collection framework (LLM-Find) that enables an LLM to autonomously select data sources and download the required geospatial data through code generation, execution, and self-debugging.[8] This framework introduces a two-tier structure consisting of a "data source index" and source-specific "handbook repositories", so that the LLM interprets the request, selects an appropriate source, and then consults source-specific access protocols, authentication procedures, and example code in order to retrieve the actual data. The authors implemented a prototype in the form of a QGIS plugin and a Python program, and used case studies involving heterogeneous data sources such as OpenStreetMap (OSM), the U.S. Census, ESRI World Imagery, OpenTopography, OpenWeather, and NYTimes COVID-19 to demonstrate that consistent automatic collection is possible. This is evaluated as a pioneering attempt to structurally formalize the "data search and acquisition" layer in the discourse on autonomous GIS.[8]

2.3 MCP

One of the recent notable trends in LLM tool integration and agent orchestration is the MCP proposed by Anthropic (Fig. 1). MCP is an open standard designed to connect AI systems, including LLMs, with external data sources and business tools, replacing the previous approach of implementing separate connectors for each system with integration through a single protocol.

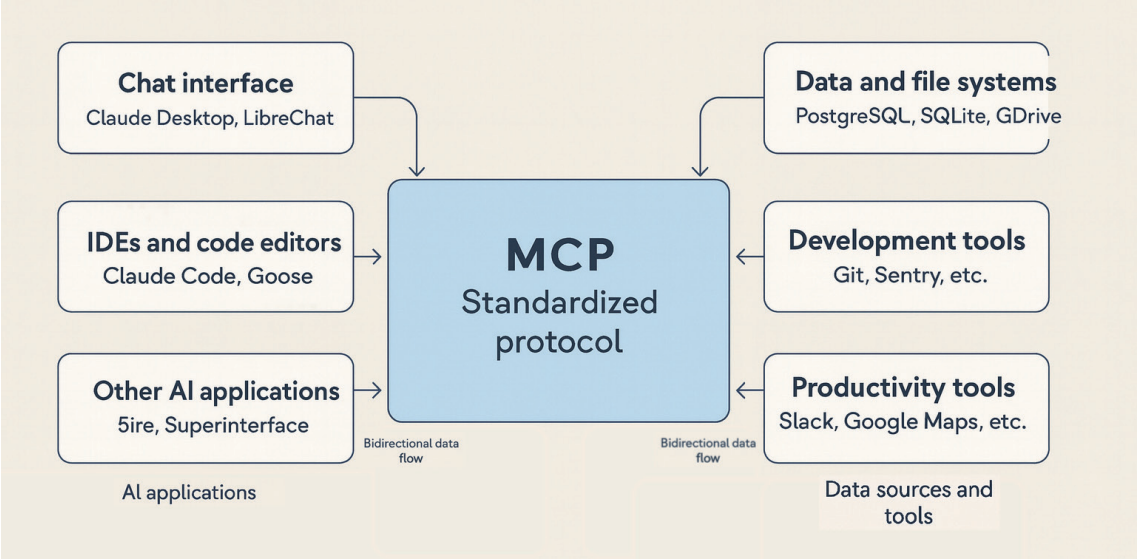

Fig. 1. (Color online) MCP.

Within a unified architecture, developers can configure MCP servers that expose data and MCP clients that connect to multiple servers, thereby providing models with various capabilities—such as file reading, function execution, and contextual prompt provision—in a standardized manner. In its official announcement, Anthropic likened MCP to a "USB-C port for AI", stating that its goal is to integrate heterogeneous resources—including local file systems, databases, cloud services, and business applications—under a single interface. Alongside its initial release, Python and TypeScript SDKs, example servers, and reference implementations for major systems such as Google Drive, Slack, GitHub, Postgres, and Puppeteer were provided, and as MCP has since been adopted by various tools including integrated development environments (IDEs), code assistants, and enterprise-embedded agents, it is reported to be emerging as a de facto standard protocol for LLM access to tools, data, and workflows.[3]

2.4 A case of optimizing large-scale MCP tool selection through RAG-MCP

Through RAG-MCP, we demonstrate a method for improving LLM tool selection accuracy and reducing prompt bloat in environments where thousands of MCP tools coexist. The key idea is that, for each user query, only the most semantically relevant MCP tools are selected from an external index via semantic search, and only their schemas are injected into the LLM while the rest are excluded, thereby simultaneously reducing the decision-making burden and token consumption. In this way, the research structurally mitigates the problem whereby "the more tool descriptions are injected at once, the harder the selection becomes and the more errors increase".[4] As shown in Fig. 2, RAG-MCP adds a tool-retrieval step before MCP-based tool execution, so that only the relevant tool schemas are injected into the LLM prompt.

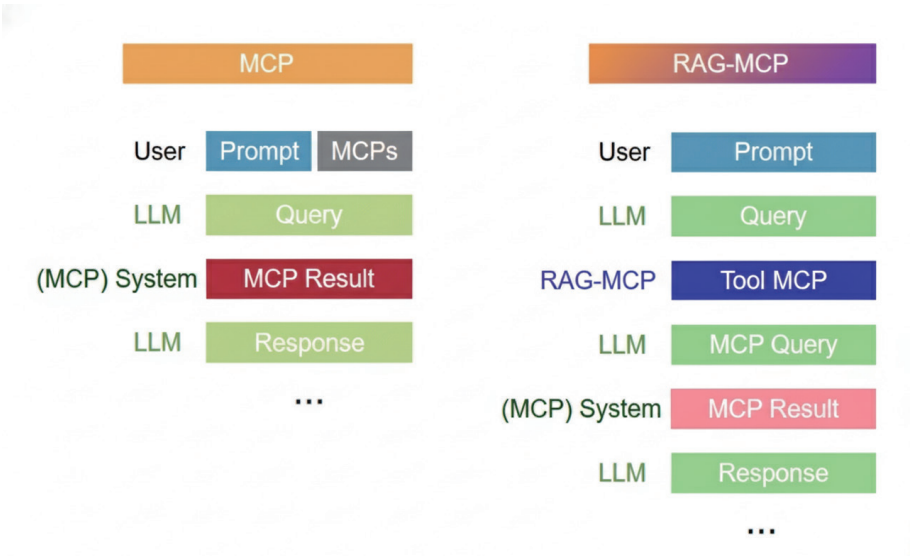

Fig. 2. (Color online) MCP and RAG-MCP during inference.

2.5 Agent frameworks for tool orchestration

LangChain and LlamaIndex are representative general-purpose agent frameworks that support tool selection and orchestration in LLM-based systems. These frameworks commonly implement an agentic workflow in which natural-language queries are interpreted, appropriate tools are selected, execution results are incorporated, and subsequent steps are determined iteratively.

In particular, LangChain provides a flexible orchestration architecture centered on repeated tool invocation and agent execution, enabling the construction of complex multistep workflows. LlamaIndex, in contrast, places greater emphasis on context-augmented agents by tightly integrating external data sources and RAG mechanisms. Together, these research and development efforts indicate that tool orchestration has already become a widely adopted design pattern in LLM-based systems, providing a general foundation for building autonomous or semi-autonomous agent frameworks.[9,10]

2.6 Maritime traffic-specialized LLM and agent studies

In this study, we summarized and organized a multimodal integrated reasoning navigation support system that combines LLMs and MLLMs specialized for the maritime traffic domain to simultaneously process AIS maritime chart videos and forward-looking camera navigation videos, and to support decision-making that takes into account navigational regulations such as International Regulations for Preventing Collisions at Sea (COLREGs). The research proceeds from the premise that the maritime environment contains a large amount of domain-specific information that differs from general data, and that existing systems remain focused on data-fusion-based situation awareness and thus fail to connect to comprehensive reasoning grounded in regulations and expert knowledge. Accordingly, it proposes a framework that reinforces semantic interactions across multiple modalities and extends to reasoning-based action recommendations that leverage legal rules and expert knowledge.[11]

Recently, domain-adaptive LLM agent studies have also been proposed to support natural-language-based situation awareness and decision support in vessel traffic services (VTS) environments. For example, VTS-LLM formulates risk-vessel identification as a knowledge-augmented Text-to-SQL task and validates its performance across diverse query styles (e.g., command style, operational style, and formal natural language) by constructing a domain-specific corpus and a query–SQL benchmark. In addition, by incorporating maritime domain knowledge injection, relational reasoning, and a query-rethink mechanism, we reported improved performance over general-purpose LLMs and SQL-centric baselines, thereby demonstrating the potential of natural-language interfaces and real-time decision support for VTS operations.[12]

2.7 Commercial maritime AI systems

AI-based systems to support maritime operations are also increasingly adopted in commercial settings.[13–15] Windward provides operational support based on agentic workflows, enabling risk analysis and compliance responses by fusing multisource maritime data into a Maritime AI platform.[13] In addition, Fujitsu has highlighted the potential of maritime AI agents to optimize operations, improve safety, and support regulatory compliance in the maritime domain.[14] Moreover, Fujitsu has reported a case study on applying an AI-based vessel-collision-risk prediction technique in the VTS context of the Singapore Strait.[15]

3. Design and Implementation Framework of EOTS-AI Agent

Figure 3 shows the overall architecture of the proposed EOTS-AI Agent framework. The framework is designed with an EOTS AI Agent Side on the left and an EOTS Side on the right, and the two areas communicate via an MCP client–server connection. First, the operator inputs a natural language query (Step 1), and the LLM interprets this query and selects the tools and workflows corresponding to the required functions (Step 2). In this process, the Tool Retrieval module queries the list of tools and schemas registered on the MCP server and injects them into the LLM's system prompt, thereby enabling the LLM to explicitly recognize the space of tools available for selection.

For the selected tool calls, verification and feedback (Step 3) are performed, including schema consistency checking and safety inspection, and during execution, the Run-time Monitoring & Self-correction module monitors the response format and exceptional situations and supports the LLM in readjusting parameters when necessary. Finally, the confirmed tool calls are delivered to the MCP server via the MCP client, where the actual tool execution takes place (Step 4), and the execution results are then returned through the MCP client–LLM path and presented to the operator in the form of natural language explanations, summaries, and suggestions for follow-up actions (Step 5).

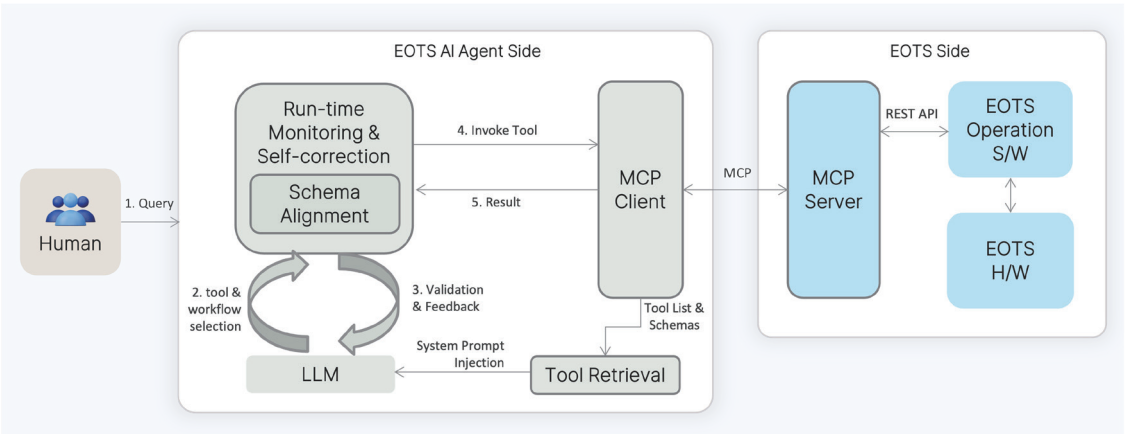

Fig. 3. (Color online) Framework of EOTS-AI Agent.

3.1 Tool retrieval module

In the proposed framework, the Tool Retrieval module functions as a tool catalog and alignment layer that mediates between the LLM and the MCP-based tool set. First, the module periodically collects and normalizes the list of tools and input schemas provided by the MCP server, maintaining an internal catalog that includes each tool's name, argument structure, and purpose (e.g., status query, control command, diagnostic function, etc.). At the start of a session, this catalog is inserted into the system prompt in the form of a summarized JavaScript object notation (JSON) schema, so that the LLM interprets queries with an explicit awareness of "which tools are currently available and with what kinds of arguments they can be invoked". Through this mechanism, while analyzing the user's natural-language instructions, the agent can autonomously plan which of the available tools to map to the functions required for EOTS operation and how to construct the arguments with which to call them.

3.2 Run-time monitoring and self-correction module

This module is responsible for closed-loop control before and after the execution phase. Before tool execution, it prevents structural errors by pre-validating missing parameters, format, range, and units of commands through the schema alignment module described in Sect. 3.3. After execution, it summarizes and organizes the success/failure information, key metrics, and error messages returned by the EOTS and feeds them back to the model, thereby inducing parameter reset, retry of corrected commands, and progression to subsequent steps. Through this cyclical procedure of "pre-consistency verification → execution → result summarization and feedback → modification and re-execution if necessary", the module systematically improves operational stability and success rate.

3.3 Schema alignment module

This module serves to verify the correctness of the tool calls generated by the LLM before they are actually executed. Specifically, it checks

(i) whether any required parameters are missing,
(ii) whether the data types of the values and any enum values to be selected match the schema,
(iii) whether the values fall within the allowed ranges (minimum/maximum) defined in the schema, and
(iv) whether units such as angles and speeds are expressed consistently and whether the notation format is valid.

If any issues are detected during this inspection process, it briefly summarizes which items are missing and what formats, units, and ranges are required, and sends this information back to the LLM to induce it to rewrite the command within the same session. By separating verification and correction into this pre-execution stage, structural errors can be blocked in advance, thereby improving the completeness and consistency of tool calls.

4. Case Studies

To demonstrate the capabilities of the proposed EOTS-AI Agent, maritime surveillance missions were provided to the agent in the form of natural-language queries. The test cases were designed by the authors with practical field experience, focusing on queries required for EO/IR-based target detection, tracking, and identification. Figure 4 shows an example of the EOTS-AI Agent GUI used in the experiments: on the left, an EO/IR surveillance video is displayed, and on the right, an interactive interface is presented where the operator inputs queries in natural language and the agent reports responses and control status. It is noteworthy that no separate advanced prompt-engineering techniques are required to design task requirements. Even users who have expertise in the maritime surveillance domain but no prior experience in using LLMs can configure the required surveillance and control tasks using only the natural-language expressions they ordinarily use, as shown in Fig. 4. These example cases were classified into three levels—Basic, Intermediate, and Advanced—on the basis of the degree of automation difficulty and the scope of work.

4.1 Basic-level tasks

Basic-level tasks are generally composed of a single command, and the procedure carried out by the agent can be summarized in four steps: natural language command interpretation → selection of appropriate tools → parameter application → verification of equipment operation. Experiments at this stage were conducted on basic operations required for EO/IR sensor operation, and whether the natural language commands were executed as intended was evaluated. A representative list of specific tasks is summarized in Table 3, and all test cases are attached in the appendix.

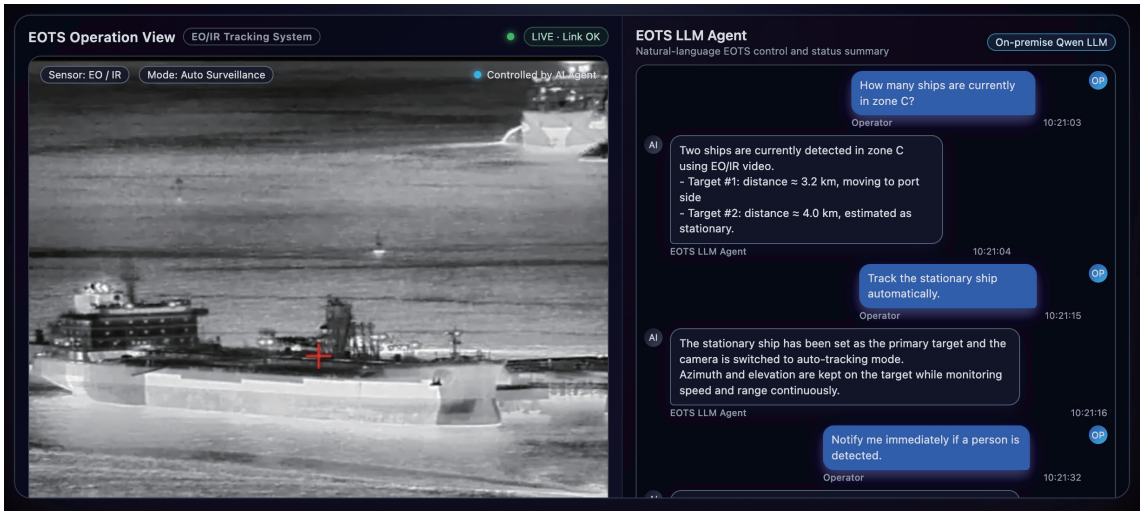

Fig. 4. (Color online) EOTS AI Agent GUI.

Table 3
Partial list of basic-level tasks.

No.	Query
1	Switch to thermal mode
2	Switch to day mode
3	Switch to SWIR mode
4	Zoom EO camera to 3×
5	Zoom IR camera to 5×
6	Switch IR camera to black-hot
7	Switch IR camera to white-hot
8	Start day camera stabilization
9	Stop day camera stabilization
10	Start thermal camera stabilization

In general, basic-level tasks consist of the most basic functions in EOTS operation, such as zoom adjustment, pan/tilt control, autofocus, automatic detection/tracking, and stop commands. The evaluation criterion judged a case as successful when the requested command was executed normally and the equipment showed the expected behavior, and as a failure when abnormal behavior occurred or the requested behavior was not performed.

4.2 Intermediate-level tasks

Intermediate-level tasks consist of composite control operations executed by coordinating two or more tools. To accomplish these tasks, the agent receives concrete, sequential instructions such as "switch mode and then start autoscan", "move to a specified bearing and then adjust zoom", or "activate autotracking mode after autodetection". The list of such tasks is summarized in Table 4, and all test cases are provided in the appendix. In particular, sensor mode switching is represented by queries such as "Switch to thermal mode then zoom IR camera to 5×" and "Switch to day mode then zoom EO camera to 3×", while azimuth/elevation motion is represented by queries such as "Rotate left 20 degrees then tilt up 5 degrees", "Rotate right 30 degrees then tilt down 3 degrees", and "Move to azimuth 30 degrees then move to elevation 3 degrees". In addition, the set includes multi-step commands that frequently appear in EOTS operating procedures, such as distance measurement ("Start distance measurement then show target position") and automatic scan-based surveillance ("Show autoscan list then start autoscan"). Even though the stepwise procedures are explicitly specified, the agent must still select the appropriate tool for each step and execute them in the prescribed order. A task is considered successful if all steps are executed correctly in the intended sequence and the final outcome matches the user's request; it is judged as a failure if any abnormal behavior occurs at an intermediate step or if the final result differs from the requested outcome.

4.3 Advanced-level tasks

Advanced-level tasks are not simple controls that end with a single instruction, but presuppose a composite surveillance and control workflow composed of multiple procedures. At this level,

Table 4
Partial list of intermediate-level tasks.

No.	Query
1	Switch to thermal mode then zoom IR camera to 5×
2	Switch to day mode then zoom EO camera to 3×
3	Start day camera stabilization then run day autofocus
4	Start thermal camera stabilization then run thermal autofocus
5	Rotate left 20 degrees then tilt up 5 degrees
6	Rotate right 30 degrees then tilt down 3 degrees
7	Move to azimuth 30 degrees then move to elevation 3 degrees
8	Increase pan speed then increase tilt speed
9	Turn on thermal camera then switch to thermal mode
10	Turn on day camera then switch to day mode

the user provides only high-level goals in natural language, such as "If there is any vessel near the left red lighthouse, automatically find it and keep tracking it.", "Scan the entire coastline and automatically focus tracking on any vessel you detect.", "Find a target near the breakwater and report its exact coordinates.", "Patrol the lighthouse area and automatically start recording when movement is detected.", as presented in Table 5, without specifying the detailed execution procedure (i.e., which tools to invoke and in what order). The full list of queries is summarized in Table 5. In this stage, the agent must infer the user's intent, select appropriate tools, and autonomously organize the necessary control commands and follow-up actions to design and complete the entire task pipeline. The evaluation criterion is the consistency at which this autonomous design satisfies the user's intended surveillance objectives and operational procedures.

4.4 Performance evaluation

All experiments in this study were conducted on a single NVIDIA A100 80 GB GPU. The LLM was served using vLLM (an OpenAI-compatible server) with Qwen/Qwen3–14, and the inference hyperparameters were fixed to temperature = 0.2 and max_tokens = 1024.

In this environment, we evaluated the performance of the EOTS-AI Agent by running a total of 80 prompt-based experiments, consisting of the representative scenarios introduced in Sect. 3 and additional scenarios. The experimental tasks comprised 50 basic-level tasks, 20 intermediate-level tasks, and 10 advanced-level tasks. Each task was allowed up to three retries with the same prompt; if the required procedure was not fully completed within three attempts, the case was classified as a failure. Overall accuracy was computed as the proportion of successful cases.

Table 6 summarizes the distributions of successes and failures across the three task levels. All 50 basic-level tasks were unsuccessful succeeded on the first attempt, yielding a 100% success rate. Likewise, all 20 intermediate-level tasks were completed successfully, with only one case requiring a single retry, resulting in a 100% success rate. In contrast, among the 10 advanced-level tasks that required scenario-based autonomous planning, the overall success rate was only 50%. This indicates that while the proposed framework operates reliably for basic- and

Table 5
Full list of advanced-level tasks.

No.	Query
1	If there is any vessel near the left red lighthouse, automatically find it and keep tracking it.
2	Scan the entire coastline and automatically focus tracking on any vessel you detect.
3	Continuously monitor this area and notify me immediately when any object appears.
4	Find a target near the breakwater and report its exact coordinates.
5	Perform full-area surveillance and continuously track the most suspicious target.
6	Check if there are multiple objects in the current view and prioritize observing the closest one.
7	Perform long-range surveillance and automatically stabilize the image when necessary.
8	Search for targets in this area and keep tracking the first one you detect.
9	When an object of interest is detected, automatically capture a snapshot.
10	Patrol the lighthouse area and automatically start recording when movement is detected.

Table 6
Performance evaluation of EOTS AI Agent Framework.

Type	Total count of tasks	Count of successful cases			Count of failed rate	Overall success cases (%)
		First attempt	Second attempt	Third attempt		
Basic level	50	50	0	0	0	100
Intermediate level	20	19	1	0	0	100
Advanced level	10	5	0	0	5	50

intermediate-level tasks, there remains substantial room for improvement in handling more complex advanced-level tasks.

To further analyze the causes of failure in advanced-level tasks, we categorized the observed behaviors at the task level into two non-overlapping types.

(i) noncompliance: where the agent failed to withhold execution or request clarification under ambiguous user goals and instead proceeded with generic or assumption-based actions
(ii) wrong tool selection: where the agent misinterpreted a goal-only instruction and selected insufficient or inappropriate tools, resulting in incomplete task execution

Table 7 summarizes the success or failure outcome of each of the ten advanced-level scenarios together with the corresponding error taxonomy. Among the ten tasks, five were successfully completed, while the remaining five failed owing to wrong tool selection. Notably, no failures were caused by execution-time schema violations, as such errors were effectively mitigated by the schema-alignment and pre-execution verification mechanisms.

The EOTS-AI Agent showed generally strong performance on basic- and intermediate-level scenarios, but the following issues were identified for advanced-level scenarios. These findings suggest concrete research directions for improving the framework's reliability in real operational settings.

Enhancing RAG over tool documentation: By vectorizing EOTS operation manuals and API references, the system can retrieve preconditions and parameter examples with higher priority, and inject procedural rules and real usage examples into the prompt.

Domain fine-tuning: By fine-tuning a Qwen-based model on EOTS operation logs, domain knowledge can be internalized in the model, thereby improving the accuracy of command interpretation and tool selection.

Table 7
Task-level success/failure outcomes and error taxonomy for advanced-level scenarios.

No.	Advanced-level query	Error taxonomy	Description
1	If there is any vessel near the left red lighthouse, automatically find it and keep tracking it.	Success	The framework correctly selected the tools appropriate for the request.
2	Scan the entire coastline and automatically focus tracking on any vessel you detect.	Success	The framework correctly selected the tools appropriate for the request.
3	Continuously monitor this area and notify me immediately when any object appears.	Wrong Tool Selection	The framework enabled monitoring but failed to execute the required alert/notification action.
4	Find a target near the breakwater and report its exact coordinates.	Wrong Tool Selection	The framework moved to the breakwater and fired LRF, but omitted the required autodetection step needed to find the target before coordinate measurement.
5	Perform full-area surveillance and continuously track the most suspicious target.	Success	The framework correctly recognized that "most suspicious" target prioritization is unsupported and selected appropriate tools without claiming unsupported behavior.
6	Check if there are multiple objects in the current view and prioritize observing the closest one.	Success	The framework correctly checked the current view, found no detected objects, and therefore did not claim unsupported closest-target prioritization.
7	Perform long-range surveillance and automatically stabilize the image when necessary.	Wrong Tool Selection	The framework enabled stabilization correctly, but failed to select additional tools such as zoom or azimuth adjustment that were expected to support long-range surveillance.
8	Search for targets in this area and keep tracking the first one you detect.	Success	The framework correctly selected the tools appropriate for the request.
9	When an object of interest is detected, automatically capture a snapshot.	Wrong Tool Selection	If the definition of an object of interest was not provided, the framework should have asked for the criteria first; proceeding with generic detection and a manual snapshot was incorrect.
10	Patrol the lighthouse area and automatically start recording when movement is detected.	Wrong Tool Selection / Noncompliance	The framework proceeded with generic detection and a manual snapshot despite the absence of a clear definition of an object of interest; a more appropriate response would have been to withhold execution or request clarification.

Such domain-specific knowledge augmentation is expected to structurally reduce incorrect tool choices, parameter configuration errors, and missing steps in procedures, and in particular to significantly improve the success rate and stability for complex operations and advanced-level tasks.

5. Conclusions

In this study, we proposed an EOTS-AI Agent framework that operates EO/IR sensors based on natural-language queries in a maritime surveillance mission environment. The proposed system connects to existing EOTS operation software through a locally deployed server-side

LLM (Qwen Instruct with vLLM runtime) and an MCP-based tool integration layer, and systematizes a series of procedures: interpretation of natural-language queries → selection of appropriate tools → parameter configuration → verification of execution results. Through this, it was shown that operators can perform EO/IR sensor control and surveillance sequences using only natural-language instructions, without directly handling detailed control commands or API specifications.

The performance evaluation was conducted on a total of 80 prompts in a single NVIDIA A100 (80 GB) GPU environment. The prompt consisted of 50 basic-level tasks, 20 intermediate-level tasks, and 10 advanced-level tasks. As shown in Table 6, both the basic- and intermediate-level tasks successfully completed all required procedures, achieving a 100% success rate, whereas the advanced-level tasks achieved a 50% success rate due to failures in tool composition and procedural design in some cases. This indicates that the proposed framework is sufficiently practical for basic sensor control and simple sequence execution, but a conservative interpretation is still necessary for high-complexity autonomous scenarios.

Several limitations were also identified during the study. First, for some prompts, schema-level errors such as missing parameters or inconsistencies in units and value ranges occurred, but these were addressed by strengthening the system prompt and enhancing the MCP tool descriptions. Second, in situations such as the advanced-level tasks, where only the goal is given and the procedure is not specified, the LLM's ability to fully and autonomously design and execute the sequence of tool selections and ordering was limited.

This study is meaningful in that it presents a concrete case showing that an open-weight LLM + MCP-based agent architecture can serve as a practical alternative for EOTS operated in a closed-network environment. Future work will aim to systematically improve performance on advanced-level tasks by enhancing RAG over tool documentation and applying domain fine-tuning based on EOTS operation logs, thereby strengthening tool-selection accuracy and error-correction capability.

References

1 Y. Zhang, C. Wei, Z. He, and W. Yu: Int. J. Appl. Earth Obs. Geoinf. **131** (2024) 103976. https://doi.org/10.1016/j.jag.2024.103976
2 H. Li, P. Yue, H. Wu, B. Teng, Y. Zhao, and C. Liu: Int. J. Digit. Earth **18** (2025) 2510566. https://doi.org/10.1080/17538947.2025.2510566
3 Anthropic: Introducing the Model Context Protocol https://www.anthropic.com/news/model-context-protocol (accessed November 10, 2025).
4 T. Gan and Q. Sun: arXiv:2505.03275 (2025). https://doi.org/10.48550/arXiv.2505.03275
5 OpenAI: GPT-5 System Card (2025). https://cdn.openai.com/gpt-5-system-card.pdf
6 Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, C. Chen, C. Olsson, C. Olah, D. Hernandez, D. Drain, D. Ganguli, D. Li, E. Tran-Johnson, E. Perez, J. Kerr, J. Mueller, J. Ladish, J. Landau, K. Ndousse, K. Lukosuite, L. Lovitt, M. Sellitto, N. Elhage, N. Schiefer, N. Mercado, N. DasSarma, R. Lasenby, R. Larson, S. Ringer, S. Johnston, S. Kravec, S. El Showk, S. Fort, T. Lanham, T. Telleen-Lawton, T. Conerly, T. Henighan, T. Hume, S. R. Bowman, Z. Hatfield-Dodds, B. Mann, D. Amodei, N. Joseph, S. McCandlish, T. Brown, and J. Kaplan: arXiv:2212.08073 (2022). https://doi.org/10.48550/arXiv.2212.08073

7 A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, C. Zheng, D. Liu, F. Zhou, F. Huang, F. Hu, H. Ge, H. Wei, H. Lin, J. Tang, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Zhou, J. Lin, K. Dang, K. Bao, K. Yang, L. Yu, L. Deng, M. Li, M. Xue, M. Li, P. Zhang, P. Wang, Q. Zhu, R. Men, R. Gao, S. Liu, S. Luo, T. Li, T. Tang, W. Yin, X. Ren, X. Wang, X. Zhang, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Zhang, Y. Wan, Y. Liu, Z. Wang, Z. Cui, Z. Zhang, Z. Zhou, and Z. Qiu: arXiv:2505.09388 (2025). https://doi.org/10.48550/arXiv.2505.09388
8 H. Ning, Z. Li, T. Akinboyewa, and M. N. Lessani: Int. J. Digit. Earth **18** (2025) 2458688. https://doi.org/10.1080/17538947.2025.2458688
9 LangChain: GitHub repository https://github.com/langchain-ai/langchain (accessed April 1, 2026).
10 LlamaIndex: GitHub repository https://github.com/run-llama/llama_index (accessed April 1, 2026).
11 H. Park, J. Jung, H. Jang, S. Lee, S. Park, and D.-G. Choi: J. Korean Inst. Next Gen. Comput. **21** (2025) 75. https://doi.org/10.23019/kingpc.21.2.202504.007
12 S. Sun, L. Zhao, M. Deng, and X. Fu: arXiv:2505.00989 (2025). https://doi.org/10.48550/arXiv.2505.00989
13 Windward: WINDWARD MARITIME AI™ — Mission-Grade Intelligence https://windward.ai/ (accessed April 1, 2026).
14 Fujitsu: AI Agents Driving Autonomy https://global.fujitsu/en-global/technology/key-technologies/news/ta-maritime_ai_agents-20251128 (accessed April 1, 2026).
15 Fujitsu: Data and AI accelerates DX realization (Part 2) — Case studies in producing business results https://corporate-blog.global.fujitsu.com/fgb/2020-01-30/data-ai-accelerates-dx-realization-part-2-case-studies-in-producing-business-results/ (accessed April 1, 2026).

About the Authors

Ki Woon Yeun has been working at Hanwha Systems' Naval Center since 2016 and is currently pursuing a Ph.D. degree in smart city studies at the University of Seoul. He is interested in the design of AI-based maritime surveillance systems and software development. (mryeun86@uos.ac.kr)

Yun-Soo Choi received his Ph.D. degree from the Department of Civil Engineering, Sungkyunkwan University, in 1992. From 1991 to 2001, he was an associate professor at Hankyong University. Since 2001, he has been a professor at the University of Seoul, and since 2018, he has been the president of the Hydrography Society Korea. His current research interests include geographic information and S-100. (choiys@uos.ac.kr)

Appendix

Table 8
Full list of basic-level tasks.

No.	Query
1	Switch to thermal mode
2	Switch to day mode
3	Switch to SWIR mode
4	Zoom EO camera to 3×
5	Zoom IR camera to 5×
6	Switch IR camera to black-hot
7	Switch IR camera to white-hot
8	Start day camera stabilization
9	Stop day camera stabilization
10	Start thermal camera stabilization
11	Stop thermal camera stabilization
12	Thermal autofocus
13	Day autofocus
14	Pan left 20 degrees
15	Pan right 30 degrees
16	Tilt up 5 degrees
17	Tilt down 3 degrees
18	Move to azimuth 30 degrees
19	Move to elevation 3 degrees
20	Increase pan speed
21	Decrease pan speed
22	Increase tilt speed
23	Decrease tilt speed
24	Stop
25	Power on thermal camera
26	Power off thermal camera
27	Power on day camera
28	Power off day camera
29	Turn LRF on
30	Turn LRF off
31	Start distance measurement
32	Show target position
33	Start thermal image enhancement
34	Stop thermal image enhancement
35	Start day image enhancement
36	Stop day image enhancement
37	Move to latitude 37°13’56”N longitude 129°32’25”E
38	Move to left red lighthouse preset position
39	Move to right breakwater preset position
40	Get detected object list
41	Start autodetection
42	Stop autodetection
43	Start autotracking
44	Stop autotracking
45	Show autoscan list
46	Start autoscan
47	Stop autoscan
48	Start recording
49	Stop recording
50	Start capture

Table 9
Full list of intermediate-level tasks.

No.	Query
1	Switch to thermal mode then zoom IR camera to 5×
2	Switch to day mode then zoom EO camera to 3×
3	Start day camera stabilization then run day autofocus
4	Start thermal camera stabilization then run thermal autofocus
5	Rotate left 20 degrees then tilt up 5 degrees
6	Rotate right 30 degrees then tilt down 3 degrees
7	Move to azimuth 30 degrees then move to elevation 3 degrees
8	Increase pan speed then increase tilt speed
9	Turn on thermal camera then switch to thermal mode
10	Turn on day camera then switch to day mode
11	Turn on LRF then start distance measurement
12	Start distance measurement then show target position
13	Start thermal image enhancement then switch IR to black-hot
14	Start day image enhancement then zoom EO camera to 3x
15	Move to left red lighthouse preset position start autodetection
16	Move to right breakwater position then start autotracking
17	Show autoscan list then start autoscan
18	Start autoscan then start recording
19	Start recording then start capture
20	Start autodetection then get detected object list