IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 15, No. 3, June 2026, pp. 2201∼2215
ISSN: 2252-8938, DOI: 10.11591/ijai.v15.i3.pp2201-2215

Multi-agent autonomous GeoAI framework for scalable and self-improving geospatial intelligence

Kim-Son Nguyen[1], The-Vinh Nguyen[1], Van-Viet Nguyen[1], Thi-Minh-Hue Luong[1], Huu-Khanh Nguyen[2], Duc-Binh Nguyen[1]
[1]Faculty of Information Technology, Thai Nguyen University of Information and Communication Technology, Thai Nguyen, Vietnam
[2]Distance Education Center, Thai Nguyen University, Thai Nguyen, Vietnam

Article Info

Article history:

Received Nov 5, 2025
Revised May 1, 2026
Accepted May 11, 2026

Keywords:

Autonomous geographic information systems
Geospatial intelligence
LangGraph
Large language models
Multi-agent systems

ABSTRACT

Large language models (LLMs) have recently expanded the scope of automation across many application domains. In geographic information systems (GIS), however, many tasks still require specialized expertise and remain difficult for non-expert users. Recent studies have explored LLM-based geospatial analysis under a single-agent paradigm, but these early systems remain limited by weak coordination, limited error recovery, and dependence on proprietary artifacts. This study proposes multi-agent autonomous geospatial artificial intelligence (MA-GeoAI), a multi-agent architecture in which the planner, coder, validator, debugger, and knowledge agents collaborate through the LangGraph framework. The framework was evaluated on three case studies: population exposure assessment, mobility pattern analysis, and county-level mortality modeling. Unlike general-purpose multi-agent LLM frameworks, MA-GeoAI embeds spatial semantics, coordinate reference system (CRS) consistency checks, geometry validation, and operation-aware coordination directly into the control loop. Across repeated runs, all evaluated systems completed the controlled artifact contract; therefore, the analysis focuses on auditability, runtime, fallback behavior, and reproducibility rather than binary task-completion superiority.

Corresponding Author:

The-Vinh Nguyen
Faculty of Information Technology, Thai Nguyen University of Information and Communication Technology
Thai Nguyen, Vietnam
Email: vinhnt@ictu.edu.vn

1. INTRODUCTION

Spatial analysis has long been a core component of modern scientific infrastructure because it provides quantitative methods for modeling and interpreting location-structured phenomena. In this context, geographic information systems (GIS) serve as an integrated platform for spatial data management, analysis, modeling, and visualization in decision-making processes [1], [2]. Spatial data are not merely geographic records; they provide scientific evidence in many interdisciplinary domains, including environmental governance, urban planning, biodiversity conservation, disaster response, and climate adaptation. Compared with conventional tabular analysis, GIS can reveal spatial patterns that are not apparent in purely attribute-based datasets. A classic example is Snow's [3] 1854 cholera map, where mapping the deaths revealed clear cluster around contaminated water pump. Despite major advances in GIS technology, many geospatial tasks still require substantial technical expertise, especially in managing coordinate reference systems (CRS), handling projections, and ensuring topological validity. As a result, GIS remains difficult to use effectively for many non-expert users.

Recent advances in large language models (LLMs) have opened new possibilities for lowering the technical barrier of GIS use [2], [4]. In general settings, LLMs can support multi-step reasoning, code generation, and external tool use from natural-language instructions. Recent studies have therefore begun to explore LLM-based geospatial analysis. For example, GIS Copilot [5] uses GPT-4 to translate user requests into executable geospatial workflows, while Li and Ning [6] describe "autonomous GIS" as a next-generation paradigm in which the workflow is monitored and executed by an AI-powered system. More broadly, general-purpose multi-agent frameworks such as AutoGen [7] and Reflexion [8] suggest that role specialization and iterative feedback may improve the reliability of complex reasoning processes. Together, these developments support the idea that LLMs could help democratize GIS for non-expert users.

Despite recent progress in LLM-assisted GIS automation, existing approaches still exhibit three practical limitations. First, single-agent workflows often generate syntactically valid but spatially unreliable outputs, especially under CRS mismatch, topology violations, or schema drift [5], [9], [10]. Second, although general-purpose multi-agent frameworks improve task decomposition and iterative reasoning, they do not explicitly encode spatial constraints, geoprocessing-specific validation rules, or domain-aware failure recovery [7], [8]. Third, many reported systems remain difficult to reproduce because prompt settings, execution environments, random seeds, and failure-handling procedures are insufficiently documented [2].

In summary, the main research gap does not lie in code generation itself, but in the absence of an architectural mechanism that ensures generated workflows remain consistent with spatial invariants. In other words, current GIS automation lacks a domain-constrained coordination layer. This observation motivates a multi-agent perspective in which task decomposition, role specialization, and coordinated problem solving are aligned with the specific constraints of geospatial computation [11], [12]. Therefore argue that autonomous GIS should be formulated as a constrained coordination problem in which reasoning, execution, and spatial validation are jointly regulated. Because each stage of a geospatial workflow requires different forms of expertise, multi-agent specialization is particularly well suited to this setting [13], [14]. The main contributions of this study are as follows: i) propose a multi-agent framework for autonomous geospatial intelligence in which planning, code generation, validation, debugging, and governed memory are assigned to specialized agents with explicit control responsibilities; ii) introduce domain-grounded coordination mechanisms that embed spatial semantics, CRS consistency, geometry validity checks, and operation-aware constraints into the agent interaction loop; iii) present a reproducibility-oriented evaluation protocol with fixed random seeds, repeated runs, transparent execution settings, and artifact-centered logging for autonomous GIS workflows; and iv) evaluate the proposed framework across heterogeneous geospatial tasks against single-agent, tool-calling, AutoGen-style multi-agent, and rule-based baselines, and we analyze performance, recovery behavior, cost, and deployment trade-offs.

The proposed architecture leverages LangGraph [15] to coordinate five specialized agents: planner, coder, validator, debugger, and knowledge. Unlike prior approaches that assign all responsibilities to a single model, each agent in the proposed framework is responsible for a distinct operational role, which helps mitigate error propagation and improve spatial reliability. Accordingly, the novelty of this work does not lie in using multiple agents per se, but in redesigning multi-agent coordination for geospatial computation under explicit spatial constraints and validation requirements. In this sense, proposed framework should be understood as domain-constrained geospatial artificial intelligence (GeoAI) architecture rather than direct adaptation of generic LLM orchestration frameworks.

2. RELATED WORK

Recent advances in LLMs have stimulated growing interest in natural-language-driven geospatial analysis and autonomous GIS systems. Early studies conceptualized autonomous GIS as an intelligent assistant capable of interpreting user intent and executing complex geospatial workflows end-to-end, marking a departure from traditional menu-driven GIS software [5], [6], [16]. Subsequent systems showed that LLMs could generate geospatial code, orchestrate analysis pipelines, and produce visual outputs with limited user intervention [6], [10]. However, many of these approaches still rely primarily on single-agent or weakly structured architectures, in which planning, reasoning, execution, and error correction are handled within a limited coordination loop. Reported results suggest that such designs may perform adequately on simple or well-structured tasks, but their reliability often declines as workflow complexity increases or when data uncertainty is introduced [5], [10], [16]. This line of work highlights a fundamental tension between

usability and robustness, showing that natural-language-driven GIS automation is feasible but still structurally constrained by limited coordination capacity.

In parallel, the literature increasingly emphasizes the need for systematic evaluation to move autonomous GIS beyond isolated demonstrations toward reproducible scientific assessment. Benchmarks such as GeoAnalystBench exemplify this shift by providing standardized task sets and evaluation criteria for LLM-based geospatial reasoning [10]. Related conceptual work on Autonomous GIS further emphasizes that geospatial automation should be evaluated as an end-to-end workflow rather than as an isolated code-generation task [6]. These studies argue that correctness in GeoAI must be assessed holistically, encompassing not only syntactic code validity but also workflow coherence, semantic alignment, and execution reliability [5], [10]. At a higher level, this reflects a broader realization that autonomous GIS is not a single-step prediction problem but a multi-stage reasoning process with interdependent decisions. Similar observations appear across intelligent tool-use systems, where errors often arise from cascading failures rather than isolated mistakes [13], [17]. Consequently, recent benchmarking and survey efforts reinforce the view that architectural design—rather than raw model size alone—plays a central role in determining system robustness and generalizability [18], [19].

Multi-agent systems (MAS) provide a well-established conceptual foundation for addressing such complexity. Classical MAS research demonstrates that distributing responsibilities among specialized agents improves scalability, fault tolerance, and adaptability in complex environments [11], [12]. In the LLM era, these principles have been revisited and extended, with multiple surveys highlighting how role specialization and structured communication can mitigate the limitations of monolithic LLM reasoning [13], [14], [17], [18]. Empirical studies further show that agentic architectures can improve error recovery and support heterogeneous reasoning strategies [19], [20]. Infrastructure frameworks such as LangChain, LangGraph, and AutoGen operationalize these ideas by providing explicit mechanisms for agent orchestration and information exchange [7], [15]. Taken together, this literature suggests that as task complexity increases, decomposing reasoning into interacting agents may improve reliability, transparency, and recovery behavior compared with monolithic designs.

The convergence of multi-agent LLM research with GeoAI has been further reinforced by the rise of open-source LLM ecosystems. Open-source models are often associated with greater transparency, reproducibility, and cost efficiency, which are particularly important in geospatial settings involving sensitive data and long-term deployment [19], [21]. Prior studies also note that open-source ecosystems facilitate domain adaptation, model inspection, and community-driven benchmarking [17], [21]. When combined with multi-agent architectures, such systems become more modular and extensible, allowing individual agents or tools to be updated without redesigning the entire workflow [7], [15].

Similar patterns are visible in AI-supported education, where recent studies emphasize integrated workflows, explainability, learning analytics, and continuous improvement rather than isolated AI tools [22]–[24]. As such, it supports the broader principle that AI systems become more reliable when domain constraints and evaluation mechanisms are embedded into the architecture. As result, these studies demonstrate the value of multi-agent decomposition for complex reasoning and tool use. However, they are not designed to enforce geospatial invariants such as CRS compatibility, geometry validity, topology-aware processing, or schema-sensitive spatial joins. The present study builds on this general line of research but shifts the emphasis from generic agent collaboration to constraint-driven coordination for reliable geospatial automation.

3. METHOD

This section describes the proposed framework from a systems perspective. Given a natural-language geospatial request, the planner first decomposes the task into an executable dependency structure, the coder produces candidate code artifacts, the validator checks spatial consistency and output plausibility, the debugger repairs localized failures when needed, and the knowledge agent stores only validated patterns for future reuse. Figure 1 summarizes this end-to-end workflow.

3.1. Architecture overview

Multi-agent autonomous geospatial artificial intelligence (MA-GeoAI) decomposes an autonomous GIS workflow into five specialized agents coordinated through a LangGraph-driven blackboard architecture [15], [25]. The purpose of this design is not to claim novelty from role separation alone, but to operationalize geospatial reliability through explicit workflow states and validation checks that are inspectable, reproducible, and repairable. Concretely, the planner translates a natural-language request into a directed acyclic graph

(DAG) of GIS subtasks with explicit inputs, outputs, and dependencies. The coder generates executable Python for each DAG node and composes a runnable pipeline while accounting for GIS constraints such as CRS requirements, geometry-type assumptions, and computational considerations for large vector layers.

The validator performs pre-execution and post-execution checks (e.g., CRS consistency, geometry validity, schema constraints, and output sanity criteria) to reduce silent spatial errors that can pass unnoticed in purely text-based evaluation. The debugger handles runtime failures by proposing minimal patches grounded in tracebacks and validator reports, thereby localizing recovery to the failing step rather than restarting the entire workflow. The knowledge agent stores reusable patterns (successful snippets, pitfalls, and repair recipes) as structured records to support reuse across tasks via retrieval-augmented prompting, with an emphasis on curating trusted artifacts instead of accumulating unverified outputs [20], [26]. Figure 1 illustrates the LangGraph-driven blackboard orchestration through which the five agents exchange intermediate states, validation signals, and recovery actions. The CRS-consistency predicate, the localized repair operator, and the governed memory update rules are provided in section 6. These definitions are intended as implementation-level specifications rather than as a trainable learning objective or optimization model. Their role in the present evaluation is assessed through artifact checks, fallback indicators, runtime logs, and reproducibility records rather than through a separate trainable ablation model.

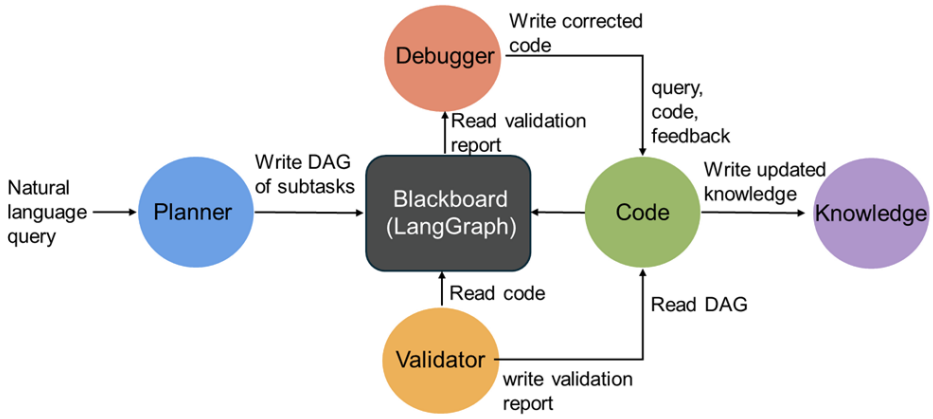

Figure 1. MA-GeoAI orchestration through a LangGraph-driven blackboard mechanism

3.2. Task decomposition: planner-generated geoprocessing directed acyclic graph

The planner converts a user query into a DAG where each node corresponds to a concrete GIS operation such as data loading, buffering, overlay, aggregation, or visualization. This decomposition reduces long-horizon brittleness by localizing errors to specific nodes, enabling partial re-execution, and making dependencies explicit for subsequent validation. Importantly, the DAG acts as an executable plan and as a validation contract: each node includes assumptions about required CRS, expected geometry types, and expected output schema, which are frequent sources of GIS failures when omitted. Figure 2 provides an example of the planner-generated geoprocessing DAG, showing how a natural-language request is translated into ordered spatial subtasks with explicit dependencies.

Figure 2. Example planner-generated DAG for population exposure analysis

The DAG representation is designed to remain inspectable by humans and compatible with re-execution [6]. This is critical in GIS because many tasks are data-centric and deterministic: a workflow that is correct should be re-runnable on the same inputs, and deviations often indicate a concrete operational issue

rather than subjective ambiguity. In addition, the DAG improves traceability by linking each output artifact to its upstream steps, which supports auditing and failure analysis during evaluation. These properties are consistent with benchmark perspectives that emphasize workflow correctness and code generation realism in geospatial automation [10], [27].

3.3. Code generation and composition: coder with retrieval-augmented prompting

Given the DAG, the coder generates Python code for each node and composes the nodes into an executable pipeline. We treat the coder's output as a high-risk artifact because small implementation errors (e.g., buffering in an unprojected CRS, overlay on invalid geometries, or schema mismatches during joins) can cascade through the workflow and produce either runtime failures or silent spatial errors. To reduce repeated mistakes, MA-GeoAI conditions code generation on retrieved context including prior fixes, validated snippets, and tool usage patterns stored by the knowledge agent [26]. This design follows evidence from code-centric agent studies showing that separating roles such as planning, coding, and reviewing can improve robustness and reduce iterative debugging cost [28], [29].

In addition, MA-GeoAI adopts conservative generation guards to reduce hallucinated tool calls and unsupported operations, motivated by hallucination and verification-oriented agent research [30], [31]. Practically, this is implemented as: i) a constrained tool vocabulary aligned with supported geospatial libraries and operations; and ii) a lightweight consistency check between requested operations and a curated set of valid geospatial functions, CRS identifiers, and geometry constraints. When the requested operation is not supported or is likely unsafe, the workflow falls back to an alternative plan step (e.g., adding reprojection or geometry repair) rather than executing an uncertain tool call. This design choice emphasizes operational validity over linguistic plausibility.

3.4. Quality assurance and error recovery: validator–debugger closed loop

Before execution, the validator enforces GIS-aware checks to reduce wasted computation and to prevent silent errors. CRS compatibility is treated as a mandatory precondition for many spatial operations, and geometry validity checks are applied to mitigate failures in overlays, spatial joins, and buffering. In particular, the minimal CRS-consistency predicate used by the validator is defined in section 6, as (1); when this predicate fails, MA-GeoAI halts downstream execution and triggers a corrective action (e.g., reprojection) before continuing. The validator also performs schema checks (field existence, join keys, and type compatibility) and post-condition checks (e.g., output non-emptiness when non-empty is expected and plausible value ranges), consistent with failure modes reported in geospatial evaluation studies [10], [27].

If runtime errors still occur, the debugger executes an iterative repair loop driven by exception tracebacks and validator context. This loop is designed to be minimally invasive: it proposes targeted patches to the failing node, re-runs only affected subgraphs when possible, and re-validates outputs before continuing downstream execution. This repair logic is conceptually related to iterative self-correction in agentic systems, but in MA-GeoAI both the trigger and the acceptance criteria are grounded in execution outcomes and geospatial validation signals rather than purely linguistic feedback [7], [8]. Each successful repair is stored as a structured record (symptom, root cause, patch, and prevention check) to reduce repeated failures and to shorten future debugging cycles [32].

3.5. Self-improvement and governance: knowledge curation with human oversight

The knowledge agent maintains a dynamic memory of validated artifacts to support reuse and continual refinement across tasks. Rather than claiming learnable parameters or theoretical optimization, we treat knowledge growth as a governed curation process: only artifacts that pass validation and execution requirements are candidates for retention, and unreliable patterns are rejected or marked as unsafe. This design choice addresses a key risk in agentic systems, namely that unverified outputs can be mistakenly promoted into long-term memory and later amplified across tasks. To mitigate this risk, MA-GeoAI supports explicit human-in-the-loop (HITL) review at well-defined checkpoints, inspired by learning-from-feedback and preference-based alignment research [33], [34].

In practice, HITL is implemented as a lightweight scoring or acceptance step for selected artifacts (e.g., final workflow, key intermediate layers, or repair patch) when the task is high-stakes or when automated checks are insufficient. The governed memory update rules and optional HITL scoring used for curation are summarized in section 6; these equations define what is logged and curated rather than introducing a learning objective. When expert time is limited, aggregated feedback mechanisms motivate collecting multiple

judgments for higher-confidence curation and for identifying systematic failure patterns [35]. During retrieval, artifacts with higher confidence—based on validation success and/or human approval—are prioritized to reduce reliance on heuristic memory and to improve accountability during iterative reuse [26].

4. EXPERIMENTAL SETUP AND RESULTS

4.1. Case studies, datasets, and reproducibility protocol

The evaluate MA-GeoAI on three case studies that reflect common, tool-based GIS workflows and cover heterogeneous data types and failure modes. As summarized in Table 1, three tasks are: i) North Carolina population exposure assessment (vector buffering and overlay); ii) France COVID-19 mobility pattern analysis (API-style time-series retrieval and visualization); and iii) United States county-level mortality modeling (tabular joins, choropleth mapping, and correlation analysis). These cases were selected to stress CRS handling, geometry validity, schema alignment, API fragility, and missing-data handling—factors that are frequent sources of silent errors in practical geospatial pipelines. The three case studies follow the proof-of-concept task design introduced in autonomous GIS [6], while the exact raw files, preprocessing artifacts, and run logs used in this study are documented in the data availability section and released through the project repository.

All experiments were executed in controlled environment on Google Colab (standard CPU, 12 GB RAM) with GeoPandas (v0.14), Shapely (v2.0), Matplotlib (v3.7), Folium (v0.14), and LangGraph (v0.2) [15]. Gemini (API) (temperature = 0) as the base model and fix library versions for all runs are used. Random seeds were set and recorded for each repetition to support exact reruns of the same pipeline. Each system-case configuration was repeated $n = 10$ times with fixed seeds (0–9), and we report means with 95% confidence intervals (CI). To decouple generation cost from execution stability, we adopted a two-phase protocol: Phase-A performed one code-generation step per (system and case) and cached the produced program, whereas Phase-B executed the cached program for all seeds without additional LLM calls unless debugging was explicitly enabled. Following benchmark-oriented practice for geospatial LLM systems, each run was treated as an end-to-end workflow instance with explicit intermediate artifacts, including the DAG, code, validation logs, and outputs, for auditing [10], [27]. This protocol was designed to improve the reproducibility, auditability, and comparability of autonomous GIS workflows across heterogeneous task settings. Table 1 summarizes the three case studies, their source datasets, task characteristics, and the dominant failure risks used to evaluate the framework under heterogeneous geospatial conditions.

The reproducibility checklist for the experiments includes the execution environment, model identifier, temperature setting, library versions, random seeds, input data paths, generated code artifacts, validation logs, fallback indicators, and recorded failure cases. These items are documented in the released repository. So that the reported runs can be independently inspected and rerun.

Table 1. Overview of the three case studies used in evaluation

Case	Data sources	Task definition (expected outputs)	Primary failure risks
NC population exposure	Hazardous-site proximity analysis in North Carolina, following the case design in [6]	Create 1-mile buffers around sites; compute tract-level and total exposed population; produce a map	CRS mismatch, invalid geometries, overlay errors, silent misalignment
France COVID-19 mobility	France mobility analysis during COVID-19, following the case design in [6]	Produce department/region-level monthly mobility change visualization (matrix + trend line + interactive December map) under a reproducible proxy policy	External endpoint instability (timeouts), schema drift, identifier alignment
US county mortality modeling	County-level COVID-19 mortality analysis in the United States, following the case design in [6]	Join mortality and demographic covariates; compute death-rate and correlation with elderly share; produce choropleth and scatter plot	Missingness, join-key drift, type mismatch, outliers, and sanity checks

4.2. Baselines and ablation settings

The compare MA-GeoAI against four baselines representing realistic alternatives for autonomous GIS automation. The first baseline, single-agent, performs planning, code generation, and execution within a single loop, matching the common LLM-Geo style used in early autonomous GIS prototypes [6]. The second baseline, ToolCall+Retries, uses Gemini (API) with structured retries and fallback policies, representing a GPT-style tool-calling baseline under same base model to control for model-family effects while isolating the contribution of orchestration and validation. The third baseline, AutoGen-MAS-sim, implements conversation-based

coordination among planner, coder, and checker roles, consistent with general multi-agent LLM frameworks [7], [36]. The fourth baseline, Rule-GIS, executes a fixed hand-specified geoprocessing pipeline without LLM reasoning, thereby isolating the contribution of automation from that of the GIS tools themselves.

Component-isolation settings also define to clarify the operational role of MA-GeoAI modules. MA-GeoAI w/o validator removes pre- and post-execution GIS checks, allowing execution to proceed without CRS, geometry, or schema gating. MA-GeoAI w/o debugger disables iterative repair and stops after the first runtime failure. MA-GeoAI w/o knowledge disables retrieval from validated cached code, forcing execution to begin from an uncurated faulty attempt. To make these ablations reproducible without additional LLM calls, we use controlled fault injection: missing-artifact faults test whether the validator prevents silent acceptance, and schema/execution faults test whether debugger and knowledge reduce unrecovered failures and fallback dependence. These settings are logged with validator-failure flags, fallback indicators, recovery flags, and external artifact audits.

4.3. Metrics and statistical reporting

This study emphasize estimation rather than significance testing by reporting means with 95% CI and effect sizes, thereby highlighting the magnitude and direction of observed differences across systems. Log-derived metrics designed are reported to reflect geospatial workflow correctness and operational reliability. Task success rate (TSR) is the percentage of runs that produce all required outputs and pass the validator's final checks: $\mathrm{TSR} = N_{\mathrm{success}}/N_{\mathrm{total}} \times 100\%$. Runtime is measured as end-to-end wall-clock seconds per run and reported as mean $\pm$ 95% CI. ErrorFlag is the fraction of runs that recorded an execution or validation error before final acceptance.

FallbackRate is additionally reported, defined as the fraction of runs that required switching to a deterministic case template to complete execution. Rubric-based code quality scoring is left as future work; therefore, the present evaluation focuses on log-derived and reproducible metrics that can be audited from the released run logs [10], [27]. For continuous outcomes, mean differences with 95% CI and standardized effect sizes are reported when variability is non-saturated across seeds. For proportion outcomes, absolute differences in proportions are reported; however, when TSR or ErrorFlag saturate under the controlled evaluation protocol, descriptive summaries are provided rather than over-interpreting null variance.

Under the controlled evaluation protocol, the evaluated systems achieved identical binary completion outcomes (TSR = 100%, ErrorFlag = 0%) across all seeds. Accordingly, the comparative analysis focuses on runtime, Phase-A generation cost, fallback behavior, and the auditability of intermediate artifacts rather than on binary task completion alone. Mean differences with 95% CI and effect sizes are reported instead of emphasizing significance tests. For continuous outcomes, a standardized mean difference (Cohen's d) are computed between MA-GeoAI and each baseline when variability is non-saturated. Because $n = 10$ is still modest, effect size estimates are reported primarily to indicate magnitude and direction, and are complemented with per-run logs and reproducibility artifacts (DAGs, logs, and outputs) in section 5.

4.4. Results by case study

4.4.1. Case 1: North Carolina population exposure assessment

This case quantifies residents within 1-mile buffers of hazardous-waste sites using tract-level population aggregation and mapping. The workflow follows buffer $\rightarrow$ overlay $\rightarrow$ aggregation $\rightarrow$ visualization, and it stresses projected CRS requirements and geometry validity for overlay operations. Figure 3 shows the resulting map artifact for the case study output. In the controlled protocol, all systems completed the required artifact contract for this case; the relevant audit signal is therefore whether completion required deterministic fallback and whether the produced artifacts passed validation.

4.4.2. Case 2: France COVID-19 mobility pattern analysis

This case retrieves and aggregates mobility indicators and produces department-level time-series and multi-panel visualizations. The workflow follows retrieval $\rightarrow$ aggregation $\rightarrow$ visualization and stresses robustness under schema drift, pagination/timeouts, and partial failures. Figure 4 presents the visualization output, which is sensitive to consistent identifiers and complete retrieval. In this case, recovery mechanisms are particularly important: failures often occur mid-pipeline, and a system that can isolate and repair the failing step avoids restarting the entire workflow.

4.4.3. Case 3: United States county-level mortality modeling

This case integrates county-level mortality with demographic covariates and reports both a choropleth map and an association plot between elderly population share and COVID-19 death rate. The workflow follows join → missingness handling → feature computation → visualization and correlation. Figure 5 shows the resulting choropleth and scatter plot. This case stresses schema alignment (join keys and types), missingness handling, and sanity checks to avoid misleading results due to incomplete merges.

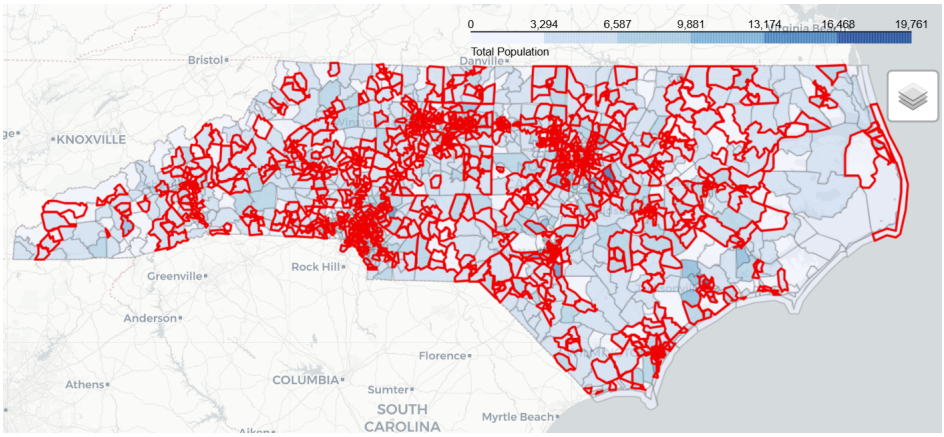

Figure 3. North Carolina map showing 1-mile buffers around hazardous sites, overlaid on census tracts

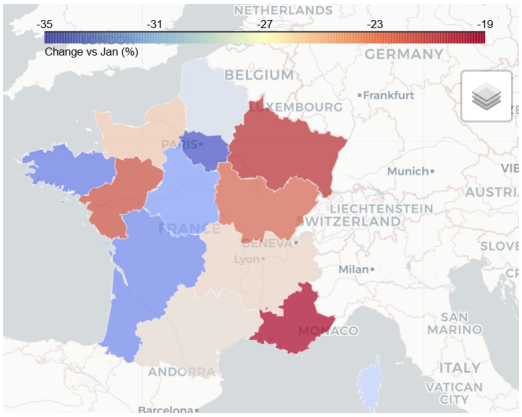

Figure 4. Multi-panel plots of mobility changes during lockdown periods in France

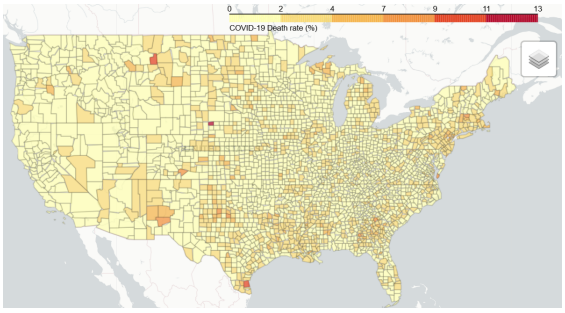

Figure 5. Choropleth of COVID-19 death rates by county and scatter plot of elderly share versus death rate

4.5. Comparative results, ablations, and cost

Table 2 summarizes aggregate performance across three case studies under the controlled evaluation protocol (5 systems $\times$ 3 cases $\times$ 10 seeds; 150 end-to-end runs). Under this protocol, all systems successfully produced the required GIS artifacts across all seeds, yielding saturated binary metrics (TSR = 100%, ErrorFlag = 0%). Consequently, the comparative analysis focuses on operational efficiency, Phase-A generation cost, fallback behavior, and the traceability of intermediate artifacts rather than on binary completion alone. For the France mobility case, a proxy/template policy was adopted to mitigate intermittent upstream endpoint failures on colab while preserving a reproducible artifact contract for evaluation. Because TSR and ErrorFlag saturate under the controlled protocol, the primary quantitative differentiators are runtime, fallback rate, and generation cost. The fallback column in Table 2 is intentionally reported because several LLM-system runs completed through deterministic templates under the controlled artifact contract. Per-run logs, generated code, validation reports, and final outputs are released to support fine-grained auditing of fallback behavior, external dependency handling, and case-specific execution traces, as discussed in section 5.

Table 2. Main results across three case studies (mean over 3 cases; $n = 10$ seeds per case)

System	TSR (%)	Runtime (sec)	ErrorFlag (%)	Fallback (%)
Single-agent	100.0	18.080 ± 0.698	0.0	66.7
ToolCall+retries	100.0	18.169 ± 0.746	0.0	66.7
AutoGen-MAS-sim	100.0	17.819 ± 1.055	0.0	66.7
Rule-GIS	100.0	17.836 ± 0.829	0.0	0.0
MA-GeoAI (full)	100.0	18.188 ± 1.218	0.0	66.7

To quantify comparative effects beyond point estimates, mean differences with 95% confidence intervals over $n = 10$ seeds per system–case configuration are reported. Because TSR and ErrorFlag saturate under the controlled protocol, the primary quantitative differentiators are per-case runtime (mean $\pm$ 95% CI), fallback behavior, and generation cost. Specifically, Phase-A token usage and call counts are reported, which capture the cost of code generation decoupled from execution stability. A consolidated summary of these cost proxies for each system is reported in Table 3, enabling transparent and model-agnostic comparison across baselines under fixed pricing assumptions. These aggregate summaries are complemented with per-run logs and reproducibility artifacts (generated code, validation logs, and outputs), which support fine-grained auditing and post-hoc analysis as discussed in section 5. Effect sizes (e.g., Cohen's d) are reported when the corresponding metric exhibits non-saturated variability across seeds; otherwise, descriptive summaries grounded in the released logs are provided to avoid over-interpreting saturated outcomes.

Table 3. Phase-A code generation cost reporting (Phase-B execution uses no additional LLM calls)

System	Phase-A tokens / calls (per system)	Notes
ToolCall+retries	671 prompt + 3067 completion / 2 calls	Phase-A only (hazardous_nc, covid_us); Phase-B no LLM
Single-agent	671 prompt + 3067 completion / 2 calls	Phase-A only (hazardous_nc, covid_us); Phase-B no LLM
AutoGen-MAS-sim	671 prompt + 3067 completion / 2 calls	Phase-A only (hazardous_nc, covid_us); Phase-B no LLM
MA-GeoAI (full)	671 prompt + 3067 completion / 2 calls	Phase-A only (hazardous_nc, covid_us); Phase-B no LLM
Rule-GIS	0 / 0	No LLM usage

Table 4 reports the controlled ablation study. The full MA-GeoAI configuration passed all external artifact and semantic checks across 30 ablation runs. Removing the validator produced a misleading reported TSR of 100%, but the external audit failed in every run because missing artifacts were silently accepted. Removing the debugger reduced audited TSR to 0% under injected execution faults because no repair path was available. Removing knowledge preserved audited TSR through the debugger, but every run required recovery and fallback to the curated template, indicating higher dependence on repair.

Table 4. Controlled ablation results over three cases and 10 seeds per case

Variant	Reported TSR (%)	Audited TSR (%)	Silent error (%)	Recovery (%)	Fallback (%)
MA-GeoAI full	100.0	100.0	0.0	0.0	0.0
MA-GeoAI w/o validator	100.0	0.0	100.0	0.0	0.0
MA-GeoAI w/o debugger	0.0	0.0	0.0	0.0	0.0
MA-GeoAI w/o knowledge	100.0	100.0	0.0	100.0	100.0

5. DISCUSSION

5.1. Key findings and implications

The experimental results are consistent with the central premise of this work: structuring GeoAI automation as a sequence of specialized, explicitly coordinated roles can improve the inspectability and auditability of multi-step GIS workflows. Rather than attributing performance gains solely to multi-agent coordination per se, the findings indicate that the key benefit arises from making geospatial assumptions, fallback events, and failure points explicit at well-defined stages of execution. In MA-GeoAI, planning, code synthesis, geospatial validation, repair, and knowledge reuse are separated into distinct responsibilities coordinated through a shared blackboard state, aligning with classical multi-agent design principles that emphasize modularity, explicit coordination, and fault isolation [11], [12].

This separation is particularly consequential in GIS settings, where many errors are not syntactic but semantic, arising from CRS mismatches, invalid geometries, schema drift, or unstable external data services. Such errors often remain silent in traditional scripting-based pipelines, leading to outputs that appear plausible but are analytically incorrect. By elevating geospatial validation and structured recovery to first-class stages in the execution loop, MA-GeoAI converts silent failures into explicit, inspectable signals and enables systematic auditing of end-to-end runs. This capability is critical for research-oriented GeoAI workflows, where reproducibility and traceability are as important as raw task completion.

A second implication concerns robustness under external data instability, which is a realistic deployment condition for tool-based GeoAI systems. In the France mobility case study, intermittent API timeouts reflect a realistic deployment condition for tool-based GeoAI systems. Rather than allowing network variability to dominate task success metrics, the framework enforces an artifact contract and records whether execution required repair or fallback. This design choice aligns with the paper's evaluation stance that correctness should be assessed through verifiable outputs and validator checks rather than through free-form textual claims, and it supports reproducible reporting of runtime and CI across repeated seeds. At the same time, the controlled evaluation protocol yields saturated binary outcomes, which limits the extent to which relative superiority can be inferred from TSR or error flags alone. For this reason, the main contribution of the evaluation lies less in demonstrating unconditional dominance than in showing how explicit validation, artifact logging, fallback reporting, and governed recovery can support reproducible and auditable GeoAI workflows under controlled execution settings.

5.2. Why role specialization helps in geographic information systems automation

From a systems perspective, the observed gains in reliability and diagnosability can be explained by how role specialization constrains error propagation. The planner externalizes the geoprocessing logic as an explicit DAG, making dependencies such as join keys, CRS transformations, and aggregation order visible and inspectable. The coder is responsible for producing executable code that conforms to a predefined artifact contract, while the validator enforces GIS-specific pre- and post-conditions, including CRS consistency, geometry validity, and artifact completeness. These checks are essential because many geospatial errors do not raise runtime exceptions but nonetheless invalidate downstream analysis.

The debugger localizes failures using structured tracebacks and validator reports and proposes minimal repairs, reducing the likelihood that recovery attempts introduce new regressions. Finally, knowledge component maintains a governed repository of validated procedures and repair patterns, which can reduce repeated failures and stabilize execution behavior across tasks [7], [8], [26], [37]. The component-isolation results support this interpretation by mapping each reliability function to an auditable log signal. Validator behavior is reflected in the gap between reported and audited TSR: without the validator, all missing-artifact faults were reported as successful but failed external audit, producing a 100% silent-error rate. Debugger behavior is reflected in the unrecovered execution-fault condition: without the debugger, audited TSR fell to 0%. Knowledge behavior is reflected in fallback dependence: without retrieval from validated cached code, audited TSR remained 100% only because every faulty first attempt required recovery and fallback. These results indicate that the contribution comes from operational validation, repair, and governed reuse rather than from agent role names alone.

5.3. Cost, scalability, and deployment trade-offs

A key deployment question is whether the benefits of role specialization justify the additional orchestration overhead. The experimental protocol separates Phase-A (one-time code generation per system–case) from Phase-B (seeded execution without further LLM calls), enabling cost accounting that is

transparent and reproducible. Under this protocol, generation cost is captured through token and call counts during Phase-A, while execution stability is evaluated independently in Phase-B. This separation clarifies that repeated experimental runs do not artificially inflate LLM usage and allows fair comparison across baselines.

In practical deployment, the relevant design trade-off is not accuracy alone, but the joint balance among reliability, execution cost, interpretability, and the amount of human oversight required to trust the output. For simple scripts or one-off analyses, the coordination overhead of a multi-agent pipeline may outweigh its benefits, and a single-agent or rule-based approach may be preferable. This observation suggests that adaptive execution policies—where the system dynamically selects between single-agent and multi-agent execution based on predicted task complexity and failure risk—are an important direction for future work [19].

5.4. Limitations and threats to validity

Several limitations should be noted. First, although the validator and debugger reduce failure propagation, they do not eliminate all error sources, particularly when external endpoints change, schemas drift, or spatial metadata is incomplete. Second, the multi-agent design introduces orchestration overhead that may not be justified for simple GIS tasks [25]. Third, performance remains dependent on the underlying language model and on the quality of intermediate validation signals [32]. These considerations suggest that the framework is most beneficial when task complexity and failure risk are high enough to justify structured coordination. An additional limitation concerns the use of deterministic fallback templates in several controlled runs. Although this decision improves reproducibility and preserves the artifact contract, it means that binary completion rates should not be interpreted as evidence that LLM-generated workflows alone solved every case. The France mobility case also uses proxy/template policy to handle intermittent upstream endpoint failures, which improves reproducibility but reduces ecological validity relative to fully live deployment conditions.

External validity is further constrained by the evolution of upstream data services. Schema changes, API deprecations, or data quality issues can affect results even when the automation logic is unchanged. Although artifact-level validation and recovery indicators mitigate these risks, no automated system can guarantee correctness without authoritative ground truth. Accordingly, MA-GeoAI is designed to support optional HITL review and governed memory updates, providing a mechanism for accountability and controlled knowledge accumulation [33]–[35].

5.5. Trust, governance, and responsible geospatial artificial intelligence automation

Because geospatial outputs may influence public health, mobility analysis, and regional planning, the main risk is not only explicit runtime failure but also silent analytical error that appears plausible to non-expert users. For this reason, the proposed system should be interpreted as a decision-support framework with reliability-enhancing safeguards rather than as an autonomous correctness oracle. Human review remains necessary for high-stakes tasks, ambiguous data conditions, and policy-sensitive outputs, especially when the consequences of spatial misinterpretation are difficult to detect automatically [30], [38].

6. OPERATIONAL DEFINITIONS USED IN MA-GEOAI ARTIFICIAL INTELLIGENCE

This section provides concise operational definitions referenced by MA-GeoAI. These expressions are not intended as a theoretical or learnable model, nor are they optimized or empirically estimated; instead, they specify implementation-level predicates and state-update rules used to structure validation, repair, and governed memory curation in a LangGraph-coordinated pipeline. The notation is consistent with section 3 and is designed to map directly to pseudocode, execution traces, and system logs.

6.1. Coordinate reference systems consistency as a precondition check

Many geospatial operations require compatible CRS to avoid spatially invalid outputs. In MA-GeoAI, CRS consistency is treated as a minimal gating predicate evaluated prior to operations that assume a shared CRS (e.g., overlay, distance-based buffering, and spatial joins). Operationally, the following binary predicat are used as in (1).

$$\mathrm{valid}(\mathrm{CRS}_A, \mathrm{CRS}_B) = \begin{cases} 1, & \text{if } \mathrm{CRS}_A = \mathrm{CRS}_B, \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

If $\mathrm{valid}(\mathrm{CRS}_A, \mathrm{CRS}_B) = 0$, the validator halts downstream execution and requests reprojection to a task-appropriate target CRS, consistent with tool-grounded correctness in autonomous GIS [6]. In

implementation, this predicate is realized as a CRS identifier equivalence check (e.g., European Petroleum Survey Group (EPSG) codes or normalized well-known text (WKT)) recorded in the validation report.

6.2. Repair operator for localized debugging

Let $c^{(t)}$ denote the current code artifact for a specific DAG node (or minimal subgraph) at repair iteration t. Let $\mathrm{err}^{(t)}$ denote the observed execution error signal, and let $\mathcal{V}^{(t)}$ denote the structured validation report. One step of the debugger-driven repair loop is operationally represented as in (2).

$$c^{(t+1)} \;=\; F\big(c^{(t)},\, \mathrm{err}^{(t)},\, \mathcal{V}^{(t)}\big) \tag{2}$$

Where $F(\cdot)$ denotes an operational patch-proposal procedure that produces a minimally modified code candidate. In practice, F is instantiated by prompting the debugger with execution and validation signals and requesting a targeted fix, followed by selective re-execution of the affected node or subgraph. This formulation captures an implementable behavior grounded in execution feedback rather than purely linguistic reflection, aligning with prior agentic self-correction patterns [7], [8].

6.3. Governed memory record for validated reuse

MA-GeoAI includes a knowledge agent that stores reusable artifacts as structured records. Let K_t denote the memory store after processing t tasks. For each completed task instance, the system records the user query q_t, the final accepted executable artifact c_t, and an automated reliability signal r_t derived from validation outcomes and execution success as in (3).

$$K_{t+1} = K_t \cup \{(q_t, c_t, r_t)\} \tag{3}$$

Here r_t is a logged quality indicator rather than a learnable parameter, capturing properties such as validation pass status, execution success, and the number of repair iterations. This representation supports retrieval-conditioned prompting by allowing future tasks to reuse previously validated patterns and repair strategies [26], [37].

6.4. Human-in-the-loop score as a governance signal

To support accountability, MA-GeoAI optionally annotates stored records with a human-provided score $h_t \in [0, 1]$ as in (4).

$$K_{t+1} = K_t \cup \{(q_t, c_t, r_t, h_t)\} \tag{4}$$

The value h_t represents an expert assessment of spatial correctness, safety, and interpretability. While inspired by learning-from-feedback and preference-based alignment literature, in MA-GeoAI this signal is used solely for governed curation and trust management rather than parameter training [33], [34]. When expert time is limited, aggregating multiple judgments can yield more stable curation decisions and reduce the risk of storing untrusted artifacts [35].

7. CONCLUSION

This study presented a multi-agent autonomous framework for auditable and reproducible geospatial intelligence. By separating planning, code generation, validation, debugging, and governed memory into specialized agents, the framework is designed to improve the reliability, reproducibility, and auditability of GIS automation under heterogeneous spatial tasks. Across three case studies and repeated runs, the evaluation shows that explicit validation, fallback reporting, structured recovery, and governed artifact reuse make autonomous GIS workflows more transparent and easier to audit. These findings suggest that domain-constrained spatial coordination is more important than generic agent orchestration alone for trustworthy GeoAI. Nevertheless, the framework remains dependent on model quality, external data stability, deterministic fallback policies, and human oversight in high-stakes scenarios. Future work will expand benchmark diversity, release richer prompt and failure traces, and strengthen HITL governance for policy-relevant geospatial decision support.

FUNDING INFORMATION

This research was supported by the University-level Scientific Research Project of Thai Nguyen University of Information and Communication Technology, grant number DH2025-TN07-06.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Kim-Son Nguyen	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	
The-Vinh Nguyen		✓				✓		✓	✓	✓	✓	✓		
Van-Viet Nguyen			✓	✓	✓					✓				
Thi-Minh-Hue Luong			✓	✓	✓					✓				
Huu-Khanh Nguyen			✓	✓	✓					✓				
Duc-Binh Nguyen			✓	✓	✓					✓				

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal Analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject Administration
Fu : **Fu**nding Acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interest.

DATA AVAILABILITY

The evaluation cases in this study follow the proof-of-concept design reported in [6]. The exact raw input files, preprocessing artifacts, proxy inputs, execution scripts, and run logs used in the present experiments are documented in the project repository at https://github.com/Nguyen-Kim-Son/langgraph. The repository provides the data references, access paths, and reproducibility artifacts necessary to rerun the reported experiments. To reduce reliance on website references in the main bibliography, raw dataset access paths are documented in the repository and are not listed as primary literature references.

REFERENCES

[1] K. J. S. Narayanan and A. Manimaran, "Recent developments in geographic information systems across different application domains: a review," *Knowledge and Information Systems*, vol. 66, no. 3, pp. 1523–1547, 2024, doi: 10.1007/s10115-023-01969-5.

[2] S. N. Kim, V. N. The, and D. B. Nguyen, "A systematic review of artificial intelligence in geographic information systems," in *International Conference on Advances in Information and Communication Technology*, 2023, pp. 20–31, doi: 10.1007/978-3-031-49529-8_3.

[3] J. Snow, "On the mode of communication of cholera," in *British Politics And The Environment In The Long Nineteenth Century*, 9th ed., London, United Kingdom: John Churchill, 2023, pp. 151–156.

[4] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*, London, United Kingdom: Pearson, 1995.

[5] T. Akinboyewa, Z. Li, H. Ning, and M. N. Lessani, "GIS copilot: towards an autonomous GIS agent for spatial analysis," *International Journal of Digital Earth*, vol. 18, no. 1, 2025, doi: 10.1080/17538947.2025.2497489.

[6] Z. Li and H. Ning, "Autonomous GIS: the next-generation AI-powered GIS," *International Journal of Digital Earth*, vol. 16, no. 2, pp. 4668–4686, 2023, doi: 10.1080/17538947.2023.2278895.

[7] Q. Wu *et al.*, "AutoGen: enabling next-gen LLM applications via multi-agent conversation," *International Conference on Learning Representations 2024*, 2024, pp. 1-44.

[8] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: language agents with verbal reinforcement learning," *37th Conference on Neural Information Processing Systems (NeurIPS 2023)*, 2023, pp. 1-19.

[9] C. Yu *et al.*, "Monkuu: a LLM-powered natural language interface for geospatial databases with dynamic schema mapping," *International Journal of Geographical Information Science*, 2025, doi: 10.1080/13658816.2025.2533322.

[10] Q. Zhang *et al.*, "GeoAnalystBench: a GeoAI benchmark for assessing large language models for spatial analysis workflow and code generation," *Transactions in GIS*, vol. 29, no. 7, 2025, doi: 10.1111/tgis.70135.

[11] M. Wooldridge and N. R. Jennings, "Intelligent agents: theory and practice," *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, 1995, doi: 10.1017/S0269888900008122.

[12] P. Stone and M. Veloso, "Multiagent systems: a survey from a machine learning perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000, doi: 10.1023/A:1008942012299.

[13] C. Sun, S. Huang, and D. Pompili, "LLM-based multi-agent decision-making: challenges and future directions," *IEEE Robotics and Automation Letters*, vol. 10, no. 6, pp. 5681–5688, Jun. 2025, doi: 10.1109/LRA.2025.3562371.

[14] X. Zhang, X. Dong, Y. Wang, D. Zhang, and F. Cao, "A survey of multi-ai agent collaboration: theories, technologies and applications," in *2nd Guangdong-Hong Kong-Macao Greater Bay Area International Conference on Digital Economy and Artificial Intelligence (DEAI)*, 2025, pp. 1875–1881, doi: 10.1145/3745238.3745531.
[15] K. Pelluru, "LangChain & LangGraph in production: architectures for multi-agent LLM systems," *Journal of Data and Digital Innovation*, vol. 2, no. 3, pp. 1–9, 2025.
[16] T. Xie *et al.*, "OSWORLD: benchmarking multimodal agents for open-ended tasks in real computer environments," *Advances in Neural Information Processing Systems*, vol. 37, 2024, doi: 10.52202/079017-1650.
[17] X. Li, S. Wang, S. Zeng, Y. Wu, and Y. Yang, "A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges," *Vicinagearth*, vol. 1, no. 1, Oct. 2024, doi: 10.1007/s44336-024-00009-2.
[18] C. Yu *et al.*, "A survey on agent workflow - status and future," in *2025 8th International Conference on Artificial Intelligence and Big Data (ICAIBD)*, 2025, pp. 770–781, doi: 10.1109/ICAIBD64986.2025.11082076.
[19] A. Koubaa, "From pre-trained language models to agentic AI: evolution and architectures for autonomous intelligence," *Preprints*, 2025, doi: 10.20944/preprints202507.1809.v1.
[20] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: interactive simulacra of human behavior," in *36th Annual ACM Symposium on User Interface Software and Technology*, 2023, doi: 10.1145/3586183.3606763.
[21] Z. Li *et al.*, "GIScience in the era of artificial intelligence: a research agenda towards autonomous GIS," *Annals of GIS*, vol. 31, no. 4, pp. 501–536, 2025, doi: 10.1080/19475683.2025.2552161.
[22] N. T. Vinh, T. N. Phung, and D. D. Cuong, "A bibliometric and thematic analysis of systematic reviews of artificial intelligence in education," *Advances in Information and Communication Technology (ICTA 2023)*, vol. 848, pp. 337–351, 2024, doi: 10.1007/978-3-031-50818-9_37.
[23] T. N. Phung, D. C. Do, T. T. Nguyen, V. S. Nguyen, T. V. Nguyen, and D. N. Le, "An integrated framework for outcome based education and AI supported blended learning in curriculum redesign and intelligent training management," *Discover Computing*, vol. 29, no. 1, 2026, doi: 10.1007/s10791-026-10088-y.
[24] T. N. Phung, "Using explainable AI to diagnose institutional inequality in student dropout across ethnic and regional groups in Vietnam," *AI and Society*, 2026, doi: 10.1007/s00146-026-03054-1.
[25] B. H. Roth, "A blackboard architecture for control," *Artificial Intelligence*, vol. 26, no. 3, pp. 251–321, Jul. 1985, 10.1016/0004-3702(85)90063-3.
[26] Y. Wang, S. Guo, and C. W. Tan, "From code generation to software testing: AI copilot with context-based retrieval-augmented generation," *IEEE Software*, vol. 42, no. 4, pp. 34–42, 2025, doi: 10.1109/MS.2025.3549628.
[27] L. Xu *et al.*, "Evaluating large language models on geospatial tasks: a multiple geospatial task benchmarking study," *International Journal of Digital Earth*, vol. 18, no. 1, 2025, doi: 10.1080/17538947.2025.2480268.
[28] M. A. Islam, M. E. Ali, and M. R. Parvez, "MapCoder: multi-agent code generation for competitive problem solving," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2024, pp. 4912–4944, doi: 10.18653/v1/2024.acl-long.269.
[29] X. Tang *et al.*, "CodeAgent: autonomous communicative agents for code review," in *EMNLP 2024 - 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 11279–11313, doi: 10.18653/v1/2024.emnlp-main.632.
[30] Z. Ji *et al.*, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, 2023, doi: 10.1145/3571730.
[31] B. Ji *et al.*, "Towards verifiable text generation with generative agent," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025, pp. 24230–24238, doi: 10.1609/aaai.v39i23.34599.
[32] G. Chen *et al.*, "AutoAgents: a framework for automatic agent generation," in *IJCAI International Joint Conference on Artificial Intelligence*, Aug. 2024, pp. 22–30.
[33] L. Ouyang *et al.*, "Training language models to follow instructions with human feedback," *36th International Conference on Neural Information Processing Systems*, 2022, pp. 27730-27744, doi: 10.5555/3600270.3602281.
[34] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," *Advances in Neural Information Processing Systems*, pp. 4300–4308, 2017.
[35] M. F. Wong and C. W. Tan, "Aligning crowd-sourced human feedback for reinforcement learning on code generation by large language models," *IEEE Transactions on Big Data*, 2024, doi: 10.1109/TBDATA.2024.3524104.
[36] S. Hong *et al.*, "MetaGPT: meta programming for a multi-agent collaborative framework," in *12th International Conference on Learning Representations (ICLR)*, 2024, pp. 1-29.
[37] H. Sun *et al.*, "Towards verifiable text generation with evolving memory and self-reflection," in *EMNLP 2024 - 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 8211–8227, doi: 10.18653/v1/2024.emnlp-main.469.
[38] C. N. Hang, P. D. Yu, and C. W. Tan, "TrumorGPT: graph-based retrieval-augmented large language model for fact-checking," *IEEE Transactions on Artificial Intelligence*, vol. 6, no. 11, pp. 3148–3162, 2025, doi: 10.1109/TAI.2025.3567369.

BIOGRAPHIES OF AUTHORS

Kim-Son Nguyen received her Bachelor of Information Technology from Thai Nguyen University of Information Technology in 2009 and her Master of Information Technology from Manuel S. Enverga University, Philippines, in 2012. Currently, she is a lecturer at the Thai Nguyen University of Information and Communication Technology, Vietnam. Her main research interests are artificial intelligence, LLMs, and GIS. In the current study, she contributed to conceptualization, methodology, software development, and original draft preparation. She is currently leading a research project on the integration of open-source language models with GIS applications for resource-constrained environments. She can be contacted at email: nkson@ictu.edu.vn.

The-Vinh Nguyen is currently a senior lecturer at the Faculty of Information Technology, Thai Nguyen University of Information and Communication Technology. He graduated with a master's degree in Information Systems Management from Oklahoma State University, United States (under scholarship 322). He completed his Ph.D. program under project 911 in 2020 at Texas Tech University, United States. His main research interests are computer vision, computer visualization, and computer in human behavior. He has authored or coauthored more than 50 publications with 16 H-index and more than 900 citations. He can be contacted at email: vinhnt@ictu.edu.vn.

Van-Viet Nguyen received his bachelor's degree in Information Technology from Thai Nguyen University in 2009 and master's degree in Information Technology from Manuel S. Enverga University Foundation, Lucena City, Philippines in 2012. He has worked as a lecturer at the Faculty of Information Technology, Thai Nguyen University of Information and Communication Technology since 2009. Currently, he is a researcher at the Thai Nguyen University of Information and Communication Technology, Thai Nguyen, Vietnam. His research interests include computer science, artificial intelligence, and communication technology. For this research, he contributed to methodology, software development, validation, and data analysis. He can be contacted at email: nvviet@ictu.edu.vn.

Minh-Hue Luong Thi received her Bachelor of Information Technology from Thai Nguyen University of Information Technology in 2010 and her Master of Information Technology from Manuel S. Enverga University, Philippines, in 2013. She has been a lecturer at the Faculty of Information Technology, Thai Nguyen University of Information Technology, since 2010. Her research interests include computer science, artificial intelligence, and communication technology. For this research, she contributed to conceptualization, investigation, validation, and supervision of the project. She has experience in managing interdisciplinary research projects combining AI and geospatial technologies. She can be contacted at email: lmhue@ictu.edu.vn.

Huu-Khanh Nguyen received his B.Sc. degree in Information Technology from Thai Nguyen University of Information and Communication Technology in 2020 and M.Sc. degree in Computer Science from the same university in 2022. He is currently a Ph.D. student at Thai Nguyen University since 2023. His main research interests are computer science, natural language processing, and machine learning applications in geospatial analysis. For this study, he contributed to methodology, software implementation, validation, and data analysis procedures. He can be contacted at email: khanhnh@tnu.edu.vn.

Duc-Binh Nguyen received the B.S. degree in Information Technology from Thai Nguyen University of Information and Communication Technology, Vietnam, in 2008, the master's degree in Information Technology from Manuel S. Enverga University Foundation, Philippines, in 2010, and Ph.D. degree in Information Engineering and Computer Science from Feng Chia University, Taiwan, in 2018. Currently, he is a vice dean and lecturer at the Thai Nguyen University of Information and Communication Technology, Vietnam. His current research interests include mobile computing, vehicle ad-hoc networks, wireless ad-hoc networks, and internet of things. For this research, he provided supervision, project administration, funding acquisition, and contributed to the overall research direction. He can be contacted at email: ndbinh@ictu.edu.vn.