Multi-Agent Framework for Automated Construction Monitoring and Documentation

Oscar Poudel
New Jersey Institue of Tenchology
University Hts, Newark, NJ 07102
op72@njit.com

Abstract

Construction sites are dynamic, complex environments that require real-time monitoring to ensure safety, progress tracking, and regulatory compliance. Traditional manual approaches are often error-prone and inefficient. This project presents an multi agentic framework that integrates large language models (LLMs) with multimodal data processing, tool-based reasoning, and contextual memory to automate construction monitoring tasks. By leveraging state-of-the-art models like LLaVA and LLaMA for image and text analysis and orchestrating agents through LangChain and Pydantic AI, the framework supports natural language interaction, risk detection, and structured documentation generation. A web-based frontend and alert system enable accessible and actionable insights for site supervisors. Experimental evaluation demonstrates moderate success in visual safety detection and strong performance in document summarization tasks. The report outlines the complete architecture, methodology, results, and opportunities for real-world deployment and further refinement.

1 Introduction

Construction sites are inherently complex environments where safety, progress, and compliance must be continuously monitored. The high-paced nature of activities, the involvement of heavy machinery, and the frequent changes in layout introduce a range of challenges, including accident risks, documentation delays, and monitoring fatigue. Despite advancements in digital construction technologies, many processes such as safety inspections, compliance logging, and issue tracking are still heavily reliant on manual observation and post-hoc reporting. These methods are labor-intensive and prone to errors and oversights with delayed or reactive responses to critical issues.

The emergence of artificial intelligence (AI), and large language models (LLMs) provides a transformative opportunity to automate and enhance monitoring workflows on construction sites. LLMs such as GPT-4, LLaMA, and PaLM exhibit advanced capabilities in reasoning, dialogue, and multimodal perception when extended with visual understanding modules like LLaVA or Flamingo. These models have demonstrated potential in domains ranging from software development to biomedical research, and are now beginning to impact architecture, engineering, and construction (AEC) industries as well. Agentic systems, where LLMs are paired with tool integration and memory systems, go a step further by enabling autonomous decision-making. Such architectures allow LLMs to plan tasks, interact with external APIs, reason over visual and textual data, and perform long-term monitoring that are essential for deploying intelligent assistants in high-stakes physical environments. Several recent studies have begun exploring the application of LLMs in civil and construction engineering. For example, Bouchard[1] demonstrated LLM use for semantic reasoning in building code compliance. Samsami[2] used GPT-4 for generating project schedules and detected significant reductions in human error. In safety-critical applications, researchers such as Nguyen[3] proposed hybrid systems that combine computer vision with AI for hazard detection, but these systems lacked contextual understanding and interactive capabilities.

This project builds on such research by developing a modular, agentic framework for construction monitoring. The system integrates LLMs with multimodal processing, retrieval-augmented generation (RAG), tool orchestration, and user interfaces to detect unsafe conditions, log documentation, and notify site personnel. This proof-of-concept aims to pave the way for real-time, autonomous safety and compliance assistants in the construction domain.

2 State of the Art and Literature Review

The use of artificial intelligence and deep learning in the construction industry has grown significantly in recent years surging in the integration of large language models (LLMs) for automation and intelligent reasoning tasks. In civil engineering contexts, LLMs are being adopted to accelerate tasks traditionally carried out by human experts. [4] researched on using ChatGPT for structural design optimization, where the model assisted in generating and evaluating multiple design alternatives. This approach improved iteration speed and design quality while enabling collaboration through natural language interfaces. In a related study, Samsami[2] demonstrated that GPT-4-based systems could be employed to automate the generation of construction schedules and sequencing plans. Their results showed a notable reduction in scheduling errors and increased efficiency in project management. Meanwhile, [1] tackled the problem of code compliance by integrating LLMs with semantic interpreters for building regulation documents for automated checks against International Building Code (IBC) clauses. Similarly, [5] explored the use of LLMs for natural language query answering over Building Information Modeling (BIM) data highlighting the role of language models in enhancing design-phase information retrieval.

Beyond text processing, multimodal AI has gained attention for visual safety monitoring. [3] developed a hybrid vision-based system to detect personal protective equipment (PPE) violations using the YOLOv5 framework. While effective in detecting physical features, their model lacked higher-level contextual reasoning. Zhou[6]proposed a multimodal knowledge graph powered by LLMs to support power grid safety, integrating image extraction and structured querying for hazard detection. Similarly, Pu[7] introduced "AutoRepo," a multimodal LLM-based system for automated construction inspections and report generation using data collected by drones, showing potential to streamline inspections and improve regulatory compliance. LLaVA [8] addressed this by combining vision with instruction-tuned LLMs enhancing models to interpret images in the context of natural language questions. Deep learning techniques have also been used for risk detection and scene classification. [9] trained CNNs to classify construction images based on safety labels and found that context-aware models outperform those relying solely on object detection. However, most of these systems lack interactivity and are not designed for autonomous action. This is where agentic LLM frameworks offer a significant advancement. The AgentBench benchmark [10] evaluated foundation models on tasks requiring perception, planning, and reasoning. The results showed that models integrated with tools and memory (e.g, LangChain agents) achieved better performance in dynamic environments, making them ideal for construction settings. Several studies emphasize the role of Retrieval-Augmented Generation (RAG) for domain-specific knowledge enhancement. Lee et al. (2024) compared RAG-enhanced GPT-4 systems with fine-tuned LLMs and found both to significantly outperform baseline models in retrieving safety knowledge from construction documents[11]. Tran[12] developed a multi-module Construction Safety Query Assistant (CSQA) utilizing LLMs to interpret safety regulation documents and deliver query-based responses to users, thus lowering non-compliance risk

Despite these advancements, there are several limitations to current approaches. First, many AI systems struggle with context awareness, especially in rapidly changing construction environments where object states and human positions evolve quickly. Second, interoperability with industry-standard platforms like BIM or Procore remains limited. Third, explainability is a major concern where users often do not trust the decision-making process of black-box models when they are used in safety-critical contexts. This project directly addresses these gaps by proposing a modular, tool-integrated, and locally deployable agentic LLM system. It builds upon LLaVA's visual understanding, LLaMA's lightweight text reasoning, and LangChain's orchestration capabilities to construct a robust pipeline for real-time safety detection and construction documentation. The integration of a retrieval-based knowledge backend (RAG) ensures that agents operate with contextual relevance, further enhancing their accuracy and reliability.

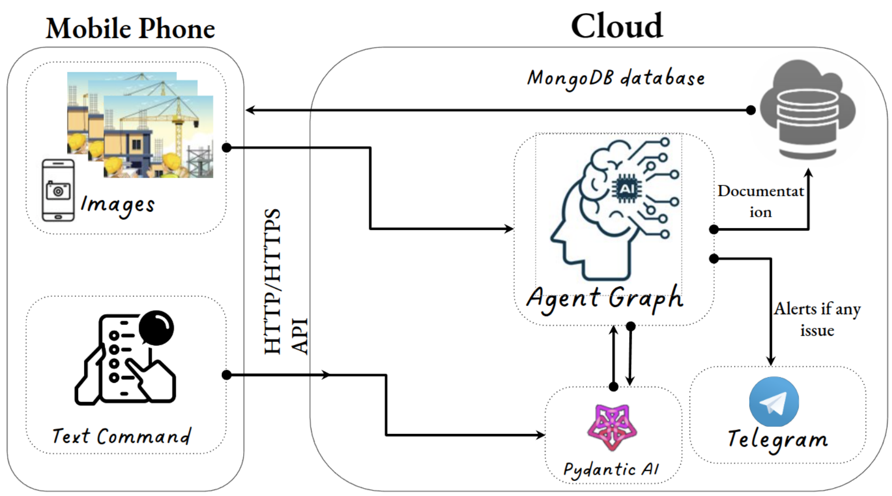


Figure 1: Overall Summary of the Architecture

3 Method

This section describes the architecture, implementation, and evaluation strategy used in developing the agentic framework for automated construction monitoring. The system is composed of a multi-agent architecture powered by large language models (LLMs), multimodal vision-language integration, contextual retrieval modules, and tool-based reasoning. The modular nature of the system allows for scalability, edge deployment, and integration with various data modalities (text and image).

This Figure 1 summarizes the pipeline components. The system begins with input acquisition in the form of text and images from a mobile device. The images are fed in the agent graph pipeline which makes decisions for the documentation or alert task. The documentation is done within the mongodb database. The alert is sent through the telegram bot agent. The final system logging is also done within mongodb. The response is then sent back to the webapp which can be visualized within the phone. The system is evaluated using the test images scraped from the web and synthetic prompts. Each of the parts is detailed in further sections of the methodology.

3.1 Model Selection and LLM hosting

To enable robust task specialization across visual and textual modalities, the system utilizes optimized set of large language models and supporting frameworks. The core design philosophy emphasizes a balance between performance and deployability, with a focus on enabling local inference to support privacy, real-time responsiveness, and edge computing. Rather than relying on cloud-based APIs that introduce latency and data privacy concerns, the framework is hosted entirely on a local Ollama[13] server, which allows for serving multiple LLMs concurrently in a lightweight, resource-efficient environment. The intent classification task is handled by the LLaMA 3.2–1B model due to its fast inference speed and low computational footprint. This model is invoked immediately after user input is received, and it performs basic classification to determine whether the query is for inspection or the documentation task. Once the intent is determined, the Controller Agent forwards the query to the appropriate downstream agent. For tasks involving visual reasoning and image interpretation, the system uses LLaVA 8B (Large Language and Vision Assistant), a vision-language instruction-tuned model capable of handling complex visual prompts. LLaVA is used by the Safety Agent to analyze site images and classify them as “safe” or “unsafe,” based on learned visual patterns. The model can also answer natural language questions about images, making it suitable for future use cases such as semantic scene understanding or construction site layout assessments. For general reasoning, structured report generation, and memory-based dialogue handling, the system uses LLaMA 3.1–8B. This model provides strong performance in multi-turn generation tasks and handles longer context lengths effectively which is ideal for the Documentation Agent. It summarizes field notes, or site reports into standardized formats that can be logged or retrieved later via a vector-based memory

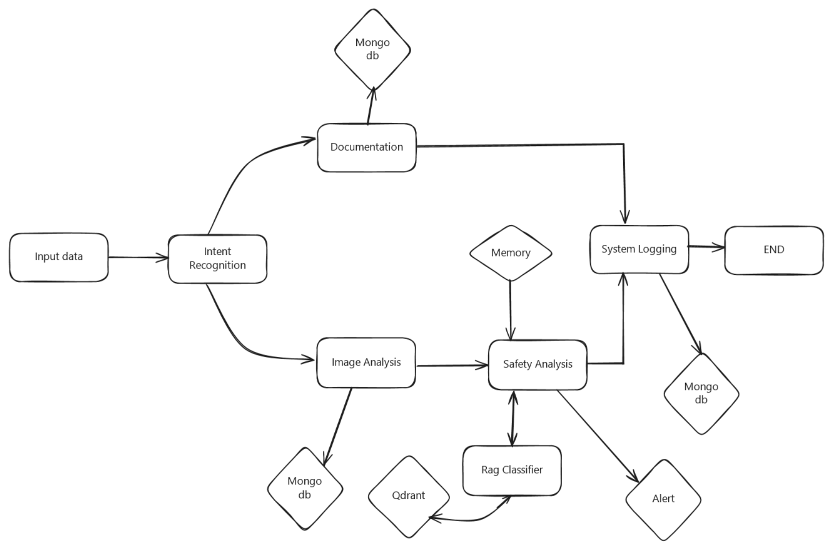


Figure 2: Agent graph for decision making and task fow

module. All models are orchestrated using the PydanticAI library [14], which provides a high-level agent framework to define tool chains, memory buffers, and control flow graphs. Agent-tool mappings, such as the link between the Safety Agent and the Telegram alert tool, are defined usingTool and AgentExecutor abstractions. To support persistent logging and reproducibility, MongoDB is employed as the system's backend log store. Every user query, intermediate agent response, tool invocation, and final output is saved along with metadata such as timestamps, latency, error codes, and model response scores.

3.2 Agent Graph and Task Flow

The core of the system's execution logic is implemented as a directed agent graph as shown in Figure 2, where each node represents an autonomous agent and edges denote task delegation. This graph-driven structure enables agents to function as collaborative workers within a shared environment. Agents communicate through predefined input/output schemas and can route control to other agents based on confidence scores, prompt content, or tool execution results. The graph begins with the Controller Agent, which uses the LLaMA 3.2–1B model to identify the query type. If the query involves image analysis, it delegates to the Safety Agent, which employs LLaVA to classify the image. Upon detection of unsafe content, this agent logs the result and calls the Alert Tool which is a Telegram bot integration, to notify construction site personnel with a brief message and annotated image. For textual tasks, such as summarizing transcribed logs or answering documentation queries, the Controller Agent routes the query to the Documentation Agent, which utilizes LLaMA 3.1–8B for structured response generation. Each agent is implemented using Pydantic AI's modular framework, with tool chaining supported by the MultiToolAgent interface. For instance, the Safety Agent accesses both the LLaVA model and the Telegram bot, while the Documentation Agent is linked to a retrieval system and a formatting utility. Agents are designed to be stateless for simplicity but can store temporary memory using a buffer window if needed. Tool execution is asynchronous and parallelizable, allowing the framework to handle multiple inputs with low latency.

3.3 Contextual Retrieval with RAG

In many scenarios, generating accurate and meaningful outputs requires background knowledge or prior examples. To meet this need, the system incorporates a Retrieval-Augmented Generation (RAG) module that augments prompts with relevant contextual documents retrieved from a vector database[15]. This is useful for tasks such as answering regulatory queries, generating formatted reports, or issuing recommendations grounded in site procedures or historical logs. The RAG

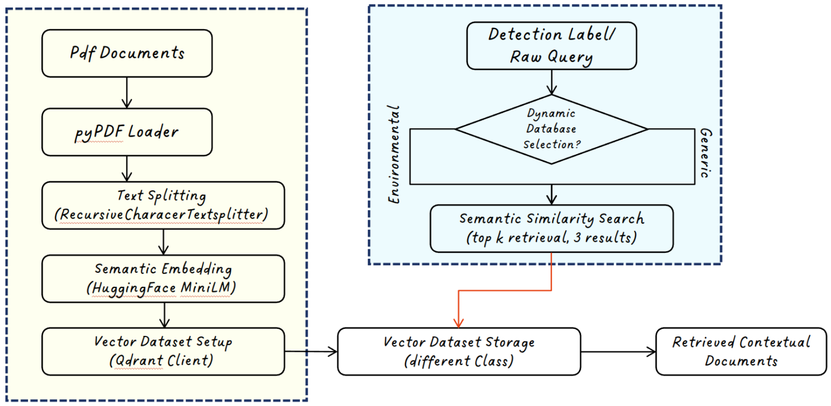


Figure 3: RAG pipeline for storing and data retrival used in the study

module as shown in Figure 3 operates by embedding documents—such as crane operation manuals, electrical safety procedures, or prior site logs—into a high-dimensional vector space using a sentence embedding model (Huggingface-embedding). Each document is preprocessed, chunked, and indexed by topic. When a query is issued to the Documentation Agent, the top-k most similar vectors are retrieved based on cosine similarity and added to the context window of the LLM prompt.

The database is organized into semantic classes (environmental safety, general construction safety) to allow fine-grained retrieval. Documents are stored in Qdrant. This contextual enrichment reduces hallucination, improves response specificity, and helps the system operate within domain boundaries—a challenge for general-purpose LLMs. Retrieved documents are also displayed in the web interface for transparency and user validation.

3.4 Safety Alert and Notification System

To translate AI-based safety detection into actionable insights (React framework[16]), the system includes a real-time alerting tool integrated with Telegram. When an image is flagged as unsafe, the Safety Agent calls the Alert Tool to sends a message containing the classification result, timestamp, and optionally a visual annotation to a designated stakeholders. This ensures that site supervisors receive immediate feedback, even when they are not actively using the web app. The Telegram bot uses the Telegram Bot API and is deployed as a lightweight Python microservice that listens for task completion messages from the agentic pipeline. It supports sending markdown-formatted alerts and images, and can be extended to support interactive commands, such as requesting justifications or logs. This real-time alerting pipeline bridges the gap between automated perception and human decision-making.

3.5 Web Interface and Deployment

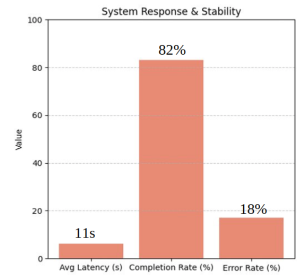
The system is wrapped in a user-facing web interface built using Streamlit for users to interact with the pipeline without coding knowledge. The interface supports uploading images, entering queries, viewing agent logs, inspecting intermediate outputs, and downloading structured reports. Results are updated in real time, and agent output confidence scores are also displayed for debugging and trust calibration.

3.6 Data Collection and Annotation

Data used for testing and evaluation consisted of site images, each manually labeled as safe or unsafe. These images were collected from publicly available construction datasets and simulation environments, along with real-world site captures. Additionally, textual prompts were created to simulate the documentation task (like construction task x% done, worker took 30 minutes break etc.) In total, 20 images were used for vision-based testing, and 10 queries were crafted for documentation-related tasks.

Metric	Value (Estimated)
Accuracy	65%
Precision	69%
Recall	75%
F1 Score	0.71
False Positives	4/20 (20%)
False Negatives	3/20 (15%)

(a)



(b)

Figure 4: (a) System monitoring classificaiton results (b) System performance evauation

3.7 Evaluation metrics

The evaluation of the agentic framework was guided by a set of well-defined metrics aimed at capturing the system's accuracy, responsiveness, and robustness. For react framework(altert testing) classification accuracy was measured by comparing the controller agent's task routing with the expected alert classificaiton. Task accuracy was evaluated using true positives, false negatives, and true negatives to determine the model's ability to correctly identify unsafe and safe conditions. The task success rate quantified how many queries were successfully completed from input to output without agent or tool failure. In addition, average latency was recorded for each query to measure system responsiveness. The pipeline completion rate was also tracked, representing the proportion of total queries that were processed successfully through all stages of the pipeline. Together, these metrics provided a comprehensive assessment of the system's functional performance.

4 Results

The performance of the proposed agentic framework was evaluated using a combination of quantitative metrics and qualitative outputs from real-time pipeline executions. The results capture both the system's analytical performance in terms of classification and response, as well as its practical utility in handling construction-related safety and documentation tasks. Figure 4(a) presents the core evaluation metrics obtained from the 20 image-based classification tasks. The system achieved an estimated accuracy of 65%, with a precision of 69% and recall of 75%, resulting in an F1 score of 0.71. This indicates that the safety detection agent is moderately effective at identifying unsafe site conditions, with a tendency to prioritize recall (identifying more unsafe cases) over minimizing false positives. Specifically, the system produced 4 false positives (20%) and 3 false negatives (15%) out of 20 image queries which shows a slight bias toward caution in safety assessments which desirable in high-risk environments.

In terms of overall responsiveness and robustness, Figure 4(b) shows the average latency per query to be approximately 11 seconds, which is acceptable for near-real-time applications in site monitoring and can be further improved with better hardware setup. The system recorded a completion rate of 82%, with an error rate of 18%, primarily due to tool invocation mismatches or input formatting issues. Despite these occasional failures, the agentic pipeline demonstrated reliable multi-agent coordination and effective fallback handling in most cases.

In addition to quantitative results, qualitative evidence of the system's operation was captured through screenshots of the web interface, MongoDB session logs, and alerting tool outputs. As shown in Figure 5, the web interface successfully guided the user through pipeline execution by allowing image uploads and natural language prompts. The resulting image analysis was routed to the vision agent, which processed the query using LLaVA and responded with a safety assessment. Documentation summaries were accurately generated and logged in MongoDB, as indicated by successful entries such as "Documentation saved successfully" and " 1st floor windows are done." Furthermore, the Telegram-based alerting tool issued real-time notifications upon detection of critical conditions. For example, one alert correctly identified a bleeding worker and issued a timestamped safety warning to a predefined communication channel. This demonstrates the practical capability of the system

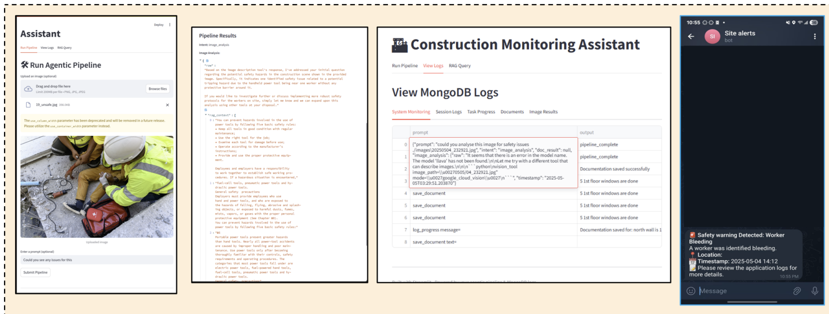


Figure 5: Outline showing the output of the pipeline

to bridge perception and action by providing intelligent, actionable alerts based on LLM-driven perception.Collectively, these results validate the proposed system's ability to detect safety risks, generate structured documentation, and autonomously communicate risks using agentic logic. While there is room for improvement in detection accuracy and tool resilience, the framework demonstrates strong potential as a foundation for intelligent construction monitoring systems.

5 Conclusion and Future Work

This project presented an agentic framework for automated construction site monitoring by integrating large language models (LLMs), multimodal vision-language analysis, contextual retrieval, and tool-based reasoning. The system was designed to detect safety hazards from visual inputs, generate structured documentation from textual prompts, and communicate critical insights in real time through integrated tools such as a Telegram alert bot and MongoDB logging interface. The modular architecture utilized lightweight and high-performance models—LLaMA for intent recognition and documentation, and LLaVA for vision-based safety detection—coordinated through LangChain-based agents and tools. This design enabled context-aware task allocation, retrieval-augmented reasoning, and multi-step workflows that mirrored real-world site monitoring operations. Through both qualitative outputs and quantitative evaluation, the system demonstrated moderate success in visual safety classification and strong performance in structured documentation generation. A completion rate of 82%, along with precision and recall rates of 69% and 75% respectively, reflect the framework's current capabilities and practical potential. Despite these strengths, several limitations were observed. The safety detection performance was occasionally hindered by low-resolution or ambiguous images, and the system exhibited sensitivity to malformed inputs and tool call failures. Additionally, while the current version performs inference locally, real-time operation on constrained edge devices may require further model compression or hardware-specific optimization. Explainability remains a challenge, particularly in justifying the LLM's decisions to non-technical users in high-risk environments.

Future work will focus on several critical areas. First, improving model calibration and fine-tuning vision-language models on construction datasets will likely enhance detection accuracy. Second, the system's resilience and fault tolerance will be improved by implementing fallback mechanisms, confidence-based agent reassignment, and retry loops for failed tool invocations. Third, real-time streaming capability and on-device deployment will be explored to support mobile robots or on-site edge boxes. Fourth, integration with additional modalities, such as audio (e.g., detecting machine alarms or worker distress calls), LiDAR, or BIM data, can provide a richer context for agent reasoning. Finally, a user-centric interface with feedback loops and interactive dialogs will be developed to increase transparency and user trust. Overall, the project demonstrates a strong proof-of-concept for LLM-driven, agent-based automation in the construction domain. By bridging perception, reasoning, and action, the system lays the groundwork for a new generation of intelligent assistants that can support safer, smarter, and more responsive construction workflows.

References

[1] K. Bouchard, M. Al-Hussein, and C. Wang. "Semantic interpretation of building code documents using large language models". In: *Journal of Computing in Civil Engineering* 37.2 (2023), p. 04023005. DOI: 10.1061/(ASCE)CP.1943-5487.0001084.

[2] Reihaneh Samsami. "Optimizing the Utilization of Generative Artificial Intelligence (AI) in the AEC Industry: ChatGPT Prompt Engineering and Design". In: *CivilEng* 5.4 (2024), pp. 971–1010.

[3] Trinh Nguyen and Amany Elbanna. "Understanding Human-AI Augmentation in the Workplace: A Review and a Future Research Agenda". In: *Information Systems Frontiers* (2025), pp. 1–21. DOI: 10.1007/s10796-025-10591-5.

[4] Nitin Rane, Saurabh Choudhary, and Jayesh Rane. *A new era of automation in the construction industry: implementing leading-edge generative artificial intelligence, such as ChatGPT or Bard*. 2024. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4681676.

[5] Ali Nakhaee, Diellza Elshani, and Thomas Wortmann. "A Vision for Automated Building Code Compliance Checking by Unifying Hybrid Knowledge Graphs and Large Language Models". In: *Design Modelling Symposium Berlin*. Cham: Springer Nature Switzerland, Aug. 2024, pp. 445–457. DOI: 10.1007/978-3-031-68275-9_36.

[6] Xiaofa Zhou et al. "Construction of a Multimodal Knowledge Graph for Power Grid Construction Safety Based on Large Language Models". In: *2024 International Conference on New Power System and Power Electronics (NPSPE)*. IEEE. 2024, pp. 21–28.

[7] Hongxu Pu et al. "AutoRepo: A general framework for multimodal LLM-based automated construction reporting". In: *Expert Systems with Applications* 255 (2024), p. 124601.

[8] Haotian Liu et al. "Visual instruction tuning". In: *Advances in neural information processing systems* 36 (2023), pp. 34892–34916.

[9] Haosen Chen et al. "Using Context-Guided data Augmentation, lightweight CNN, and proximity detection techniques to improve site safety monitoring under occlusion conditions". In: *Safety science* 158 (2023), p. 105958.

[10] Xiao Liu et al. "Agentbench: Evaluating llms as agents". In: *arXiv preprint arXiv:2308.03688* (2023).

[11] Jungwon Lee et al. "Performance comparison of retrieval-augmented generation and fine-tuned large language models for construction safety management knowledge retrieval". In: *Automation in Construction* 168 (2024), p. 105846.

[12] Si Van-Tien Tran et al. "Leveraging large language models for enhanced construction safety regulation extraction". In: *Journal of Information Technology in Construction* 29 (2024), pp. 1026–1038.

[13] Ollama. *Run LLMs locally*. 2024. URL: https://ollama.com/.

[14] Pydantic Team. *PydanticAI: A Python Agent Framework for Generative AI*. Accessed: 2025-05-11. 2023. URL: https://ai.pydantic.dev/.

[15] Yunfan Gao et al. "Retrieval-augmented generation for large language models: A survey". In: *arXiv preprint arXiv:2312.10997* 2 (2023), p. 1.

[16] Shunyu Yao et al. "React: Synergizing reasoning and acting in language models". In: *International Conference on Learning Representations (ICLR)*. 2023.