Framework for LLM-Enabled Construction Robot Task Planning: Knowledge Base Preparation and Robot-LLM Dialogue for Interior Wall Painting

Kyungki Kim *, Prashnna Ghimire *, Pei-Chi Huang

Posted Date: 17 June 2025

doi: 10.20944/preprints202506.1331.v1

Keywords: Construction Robotics; Large Language Models (LLMs); Building Information Modeling (BIM); Robot-LLM Communication; Robot Task Planning; ChatGPT

 doi:10.20944/preprints202506.1331.v1

Article

Framework for LLM-Enabled Construction Robot Task Planning: Knowledge Base Preparation and Robot-LLM Dialogue for Interior Wall Painting

Kyungki Kim [1,*], Prashnna Ghimire [1] and Pei-Chi Huang [2]

[1] Durham School of Architectural Engineering and Construction, University of Nebraska-Lincoln, USA
[2] Department of Computer Science, University of Nebraska-Omaha, USA
* Correspondence: kkim13@unl.edu

Abstract: Task planning for a construction robot requires systematically integrating diverse elements, such as building components, construction processes, user input, and robot software. Conventional robot programming complicates this by requiring precise entity naming, relationship definitions, unstructured language interpretation, and accurate action selection. Existing research has focused on isolated components, such as natural language processing, hardcoded data linkages, or BIM data extraction. We introduce a novel framework using an LLM as the cognitive core for autonomous construction robots, encompassing both data preparation and task planning phases. Leveraging OpenAI's ChatGPT4, we demonstrate how LLMs can process structured BIM data and unstructured human inputs to generate robot instructions. A prototype tested in a simulated environment with a mobile painting robot adaptively executed tasks through real-time dialogues with ChatGPT4, reducing reliance on hardcoded logic. Results suggest that LLMs can serve as the cognitive core for construction robots, with potential for extension to more complex operations.

Keywords: construction robotics; large language models (LLMs); building information modeling (BIM); robot-LLM communication; robot task planning; ChatGPT

1. Introduction

Recent advancements in automation and robotics have the potential to address persisting challenges of the construction industry, including labor shortages [1], high incidence rate of injuries and fatalities [2], and the stagnant productivity growth [3]. The adoption of autonomous robots is considered as a promising direction to address these problems by automating labor-intensive and dangerous works in construction projects [4]. In the construction industry, various commercial solutions (including robots for surveying, excavation, bricklaying, and demolition) are already demonstrating the possibility of automating a wide range of construction tasks and achieving substantial improvements [5]. However, literature suggests that achieving the full potential of construction robotization depends on developing useful robots as well as seamlessly integrating the robots into construction management processes and tools [6–8]. 4D Building Information Modeling (4D BIM) is one of the most advanced technologies widely used for construction process planning [9], safety management [10], and constructability analysis [11,12]. Several academic studies then extended the boundary of the standard 4D BIM technology for the automation of proactive safety planning [13], real-time worker safety monitoring [14], planning for temporary structures [15], and work progress monitoring [16]. The integration of 4D BIM with robot operations, particularly through robotics frameworks, such as the Robot Operating System (ROS) [17,18], presents promising direction for enhancing the effectiveness of robotization in construction. This integration can allow

 doi:10.20944/preprints202506.1331.v1

robots to plan and execute tasks more efficiently by utilizing rich construction-related information directly from 4D BIM models.

However, unlike human workers, who can intuitively understand and adapt to rough work plans, robots require extremely precise operational descriptions detailed down to fractions of a second [19]. As presented in [20], performing task planning for a construction robot involves the challenge of systematically integrating information about at least building elements, construction processes, and robot control software. This is a complicated process of accurately naming individual entities, clearly defining the relationships among them and their properties, and maintaining a high degree of precision to ensure reliable task execution [20,21]. Existing studies on BIM-based robot task planning [7,22–25] primarily rely on hardcoding the integration of heterogenous information to generate robot task plans, which requires significant software modification when new situations or commands arise. A recent study [20] attempted to combine varied types of information using a Linked Building Data (LBD)-based data modeling approach; however, linking data via names can still be considered a form of hardcoding. These conventional approaches lack flexibility, as mistakes or discrepancies in naming can lead to failures or unexpected outcomes and limit the robot's adaptability to unplanned situations. However, in construction settings, different robot behaviors can be required at various locations within a project, or workers and superintendents may request changes, complicating static data linking further. By incorporating various aspects such as safety protocols, human-robot interaction, and variable site conditions, robot task planning becomes more complex, making static data linking unsustainable at scale. Furthermore, robot task planning often requires adding new properties to building elements or creating unconventional elements in BIM. For instance, new spatial coordinates of steel beams for robotic transportation and assembly can be automatically calculated using IFC files [26], whereas elements like a material storage area for interior wall frames are manually created in a BIM model [27]. However, no existing study has comprehensively addressed how to create and integrate robot-related properties or objects into BIM through a systematic process – from developing a robot-oriented BIM model and defining robot task specifications to aligning these elements with the perspectives of end users, such as construction planners tasked with establishing plans for robot operations. Considering these deficiencies, there is a need for a framework that flexibly synthesize diverse data formats – ranging from structured files (e.g., BIM models, construction schedules) to unstructured inputs (e.g., human language instructions) – into actionable robotic task plans without hardcoding.

These challenges associated with data integration stem from the lack of knowledge and common sense of the robotic system itself that allows the flexible interpretation of data. To overcome the deficiency, our study explores the use of Large Language Models (LLMs) equipped with extensive world knowledge to seamlessly integrate diverse data and transform them into context-aware instructions for construction robots. In the field of artificial intelligence (AI), LLMs have revolutionized natural language processing (NLP) by shifting the paradigm away from reliance on narrowly focused labeled data for specific tasks. Notably, models like OpenAI's Generative Pre-Trained Transformer (GPT-4) [28] demonstrate exceptional capabilities in zero-shot and few-shot learning enabling them to understand context from minimal prompts and examples. Recent studies [29–31] demonstrated the potential of LLMs in enhancing a robot's ability to understand the operational context and adapt its actions leveraging their broad knowledge base and natural language comprehension. Our study introduces a novel framework specifically created for autonomous construction robot task planning leveraging the real-time robot-LLM communication. The proposed framework includes a set of comprehensively aligned procedures designed to form a robust knowledge base: creating a BIM model enriched with robot-related objects (e.g., painting locations with sequence, stroke time, and area properties), defining robot task specifications that incorporate information from the robot-oriented BIM model, and establishing guidelines to enable the LLM to effectively interpret and utilize the provided data. Leveraging this knowledge base, the pretrained foundation LLM, equipped with world knowledge and embedded common sense, generates task plans for the robot based on diverse construction-related and robot-related inputs –

 doi:10.20944/preprints202506.1331.v1

including project data from 3D/4D BIM, task instructions articulated in human language, and situational interactions with superintendents – with minimal reliance on hardcoding. While some hardcoding is required for data extraction and skill execution, the framework eliminates the need for extensive hardcoding of the robot's cognitive capabilities. We developed a prototype software program using Autodesk Revit and ROS [32] focusing on a mobile robot tasked to perform wall painting in a residential building. A case study in a simulated construction environment demonstrates how the "Husky Painter" robot adaptively plans its actions while communicating with ChatGPT4 without the need for hardcoded logic.

2. Literature Review

This section reviews the existing body of research relevant to the utilization of robots and LLMs in construction. The review focuses on the current methodologies in construction robot task planning with BIM, the application of LLMs within the broader construction industry, and the specific use of LLMs for enhancing robot task planning and control.

2.1. Current State of Construction Robot Task Planning Using BIM

Several studies utilized information contained in BIM models to facilitate planning and execution of construction robot operations. Lundeen et al. [22] presented a method that allows an industrial robot arm to dynamically adjust the motion plans for joint filling. This study automatically compared as-planned geometry data in a BIM model against the as-built geometric data obtained from sensors to examine deviations that can commonly occur in construction projects. Follini et al. [23] constructed a "world" – a term commonly used to describe the environment for robot simulation – utilizing BIM data. Specifically, they generated a 2D metric map from an Industry Foundation Classes (IFC) file. They used the map of the construction site for autonomous navigation safely maneuvering around areas with temporary structures. Ding et al. [24] utilized image-based 3D scene reconstruction to produce a BIM model that contains sufficient information required for robotic brick assembly. Their bricklaying robot transformed the placement points of bricks from the coordinate system of the reconstructed scene into the robot's coordinates system to execute pick-and-place. Kim et al. [7] achieved a greater level of integration of BIM and ROS frameworks for construction robot task planning. They developed a series of techniques that generates a 3D robotic world from an IFC file for robotic painting simulation. This study presented structured descriptions for robotics work tasks and environments in XML format and performed detailed task planning in simulated construction environment. A follow-up study of the authors [33] further expanded the use of BIM by creating a semantic Universal Robot Description Format (URDF) [34] representation of a building that can directly inform construction robot task planning. This study performed a case study demonstrating the use of project lifecycle information for construction progress monitoring. Chong et al. [25] streamlined robotic task planning process itself with information in a BIM model concentrating on wood frame assembly. Their approach guided robot manipulator movements based on geometric information of building elements reducing the time and manual efforts needed to create robot simulations. Yin et al. [35] introduced a method to enhance autonomous robot localization and navigation leveraging BIM models. They proposed a method that involves the conversion of original BIM data into a semantic building point cloud and alignment LiDAR point cloud data with the building point cloud using iterative closest point registration. This study reported centimeter-level accuracy in the field evaluation.

2.2. Utilization of LLMs in Construction Industry

In recent years, studies concentrated on leveraging LLMs to enhance construction processes and address problems in the construction industry. Early conceptual exploratory studies, such as Ghimire et al [36], Rane et al [37], Saka et al [38], and Taiwo et al. [39], have evaluated opportunities and challenges of LLMs throughout multiple phases of the building lifecycle – ranging from feasibility and design to procurement, construction execution, and operation and maintenance – through well-

 doi:10.20944/preprints202506.1331.v1

known approaches like zero-shot learning, few-shot learning, chain-of-thought reasoning, retrieval augmented generation, and fine-tuning. These studies also detail limitations at two levels: technological hurdles, including domain-specific knowledge, hallucinations, accuracy, generalizability, and cost; and industry-specific barriers such as acceptability, skill requirements, and regulatory challenges in construction. Other studies presented use cases of adapting LLMs to existing processes. In 2023, Zheng and Fischer [40] introduced a framework to utilize prompts to extract detailed information from comprehensive BIM models. Rane et al. [37] proposed a framework for data exchange between BIM and LLM models emphasizing interoperability, data consistency, and user-friendly interfaces. Furthermore, Saka et al. [38] developed a prototype for optimizing material selection by integrating BIM with a GPT model and tested few-shot, zero-shot, and edge case prompting. However, focusing on extracting the data of building elements, this study has not fully utilized the extensive world knowledge of LLMs for the selection of appropriate materials. Additional implementations include ChatGPT-driven scheduling by Prieto et al. [41] that logically organized activities satisfying project requirements. Similarly, Hassan et al. [42] proposed a method to identify potential safety hazards by submitting a prompt of the current situation to the BERT sentence-pair model fine-tuned with the OSAH database.

2.3. Utilization of LLMs for Robot Task Planning and Control

The use of LLMs with expansive world knowledge can be a potential solution to address the challenge of transforming diverse and heterogeneous information into appropriate robot actions. LLMs have garnered significant attention in recent years due to their remarkable capabilities in natural language processing NLP tasks. Beyond their traditional use in language generation and comprehension, researchers have explored the potential of LLMs for robot planning tasks and control, as task planners can leverage the wealth of external world knowledge embedded within these models. LLMs, such as GPT models, are large-scale neural network architectures trained on vast amounts of textual data. These models have demonstrated proficiency in various NLP tasks, including language generation, translation, summarization, and question-answering [43]. Researchers have explored the use of LLMs to represent tasks and goals in natural language, enabling intuitive human-robot interaction and task specification [44,45]. Micheli and Fleuret [46] were among the pioneers in fine-tuning LLMs for task planning. Recent LLMs studies have shifted focus towards encoding extensive world knowledge and demonstrate emerging capabilities for planning [47,48] through few-shot or zero-shot in-context learning [48–50]. Pre-trained LLMs have emerged as valuable tools for task planning in robotics and embodied agents, leveraging their vast linguistic knowledge to inform decision-making processes and action generation. Huang et al. [29] introduced Inner Monologue, which utilizes textualized environmental feedback to generate actionable tasks, whereas Shunyu et al. [51] introduced ReAct, significantly advancing the closed-loop methodology by integrating reasoning and action components, thus enhancing its real-world effectiveness. Moreover, LLMs have been utilized to solve conventional tasks and motion planning problems. Ding et al. [52] proposed LLM-GROP which utilizes a pre-trained LLM to define symbolic goals and determine continuous object placements for rearranging semantic objects, serving as input for a classical task and motion planner. Chen et al. [53] employs an LLM to transform natural-language task specifications into a formal language that can be processed by readily available task and motion planning algorithms. Wang et al. [54] introduced LLM^3, which integrates LLMs into the traditional tasks and motion planning framework, employing pre-trained LLMs to generate symbolic action sequences and determine continuous action parameters to iteratively refine motion planner.

Integrating environmental feedback during runtime has demonstrated significant enhancements in task planning as it anchors LLMs for the generation of more realistic plans [29–31]. The outputs of LLMs can be integrated with affordance functions to enhance their grounding and alignment with real-world scenarios [55]. Usually, the approaches utilizing environmental feedback for task planning can be categorized into two main groups: static and dynamic planners [56]. Static planners utilize feedback to ensure that the specified conditions in the plan align with the robot's

 doi:10.20944/preprints202506.1331.v1

environment, thereby preventing execution errors. In these situations, the feedback does not change the plans that have been generated [31,57]. Conversely, dynamic planners leverage feedback from the system to validate essential conditions and adapt the plans, thereby enhancing their execution [29,30,51,58]. Bhat et al. [56] introduced a dual-LLM system, integrating error messages and environmental data to dynamically refine task planning, the first one to implement in a realistic physics simulator and the Franka Research 3 Arm. Driess et al. [59] proposed PaLM-E models that merged PaLM-540B LLM with extra input modalities to predict feasible low-level tasks and handle failed low-level tasks. In construction, You et al. [60] evaluated the reasoning capability of ChatGPT in material stacking, Hanoi tower puzzle, and plumbing assembly tasks that can be further applied in more realistic construction simulations. Luo et al [61] utilizes LLM to generate code for construction assembly robots. Park et al. [62] introduces a framework enabling human workers to interact with construction robots through natural language instructions, specifically targeting robotic pick-and-place tasks for drywall installation. This study made significant advancements in natural language understanding and information mapping allowing necessary for communication between humans and robots.

2.4. Point of Departure

Studies have presented various possibilities of adopting robots for the automation of construction tasks. Even with the advancements, many of the studies primarily focus on extracting data from individual building elements as input for robot skills rather than addressing the challenge of synchronizing complex and unorganized information. For instance, Kim et al. [7] generated a strictly formatted XML file from a BIM model for robotic painting plans, which had to be parsed by another software in a ROS-based simulation, relying on hardcoding the alignment between BIM data generation and ROS data interpretation. Similarly, Chong et al. [25] defined a class object to transfer data about light wood frame elements for robotic pick and place control. Even though Zhu et al [20] presented a name-based data linking of diverse information required for robot task planning, the reliance on hardcoding and the lack of adaptability remain existing. Among limited works that incorporated LLMs in construction robots, Park et al. [62] utilized the natural language understanding to interpret user commands and complex geometric data of drywall panel elements for automated pick-and-place operation. While this state-of-the-art study presented a case of leveraging LLMs for construction robot task planning, it concentrated primarily on enhancing natural language understanding. Also, the pioneering study [40] on the prompt-based data extraction from BIM models focused on answering queries directly related to attributes of building elements, such as names, elevations, and dimensional properties. While the LLM-based BIM information query can be useful for robot operations, this still leaves unanswered questions about how a construction robot can handle the flow of data from multiple sources in various formats. Existing research has primarily focused on isolated components, such as improving natural language understanding [62], hardcoding specific data linkages [20], or data extraction from BIM [40], but there has been no comprehensive framework that outlines how LLMs can be systematically integrated into the task planning for construction robots. This includes the steps of preparing diverse construction-related data (e.g., building elements, schedules, safety protocols) and robot-specific data (e.g., task specifications, control parameters), presenting this information to the LLM, and enabling the LLM to generate and adjust task plans through ongoing dialogue with the robot and humans. Such a framework could serve as a foundation that enables the flexible integration of new data, the design of robot operations for new tasks, and the adaptation of different robot behaviors to varying conditions, all without requiring significant software programming.

3. Objective and Scope

To address the discussed gap, our study presents a novel framework for LLM-based construction robot task planning. The proposed framework synthesizes both structured and unstructured data into work plans for autonomous construction robots, covering a broad spectrum

 doi:10.20944/preprints202506.1331.v1

of input data, such as BIM models, construction schedules, robot task specifications in natural language, and onsite instructions from construction professionals. The framework provides a structured procedure for creating and organizing such data, functioning as a knowledge base for the robot. This knowledge base serves as the foundational information that enables the LLM to interpret the given situation and generate appropriate work plans. Another fundamental feature of the proposed framework is the dynamic and real-time conversations between robots and LLM. Throughout the operation, a robot submits prompts, such as "I am the painting robot. What is my first task today?" and receives responses, like "Your first job is to navigate to the center of the living room for wall painting" in both human language and robot-interpretable formats like JSON. This interactive approach allows a construction robot to utilize the general world knowledge of an LLM to interpret given situations and determine required actions without the significant efforts to develop extensive programming infrastructures. This reduces reliance on hardcoding numerous if-then statements and explicit coding to translate various types of data simplifying the integration of robots into dynamic construction environments. Additionally, the utilization of natural language allows construction professionals without robotics expertise to provide guidelines for robots to properly perform tasks in given situations. We develop a prototype software program and evaluate its effectiveness in a simulated construction environment. Specifically, we use the prototype to assess how a mobile painting robot communicating with ChatGPT4 can accurately understand the context of its tasks and plan its actions accordingly.

The study explicitly excludes certain areas to avoid potential misunderstandings regarding its contributions. We limit the scope of the study to focus on laying the foundational steps for embedding situational awareness and decision-making capabilities to a construction robot via the robot-LLM communication, specifically within the context of robotic wall painting. The research is conducted using a ROS-compatible robot simulator called Gazebo [63], and the deployment of actual physical robots is excluded. Robot operations in Gazebo directly use coordinates and states of objects, such as human workers, provided to the robot as known contextual information. By making work environment's details explicitly known to the robot, the scope of this study does not include sensing technologies to estimate the presence and states of objects within the construction site. While data processing by the LLM significantly reduces the amount of coding associated with the cognitive capabilities of the robot, the extraction of data from BIM and the execution of elemental robot skills still rely on hardcoding. Specifically, the extraction of data from BIM is achieved through a BIM API explicitly coded to select certain objects and their properties, which are then exported as a text file. Additionally, a robot's skill set (e.g., navigation, arm motion) cannot be generated by the LLM and therefore needs to be readily coded. Finally, Interaction with ChatGPT4 is conducted through an online Application Programming Interface (API), and the study does not extend to the use of local LLM capabilities which will be required in the future for the safety-critical implementation using real robots.

4. Automated Construction Robot Task Planning via LLM-Robot Communication

4.1. Framework for LLM-Enhanced Construction Robot Task Planning

Figure 1 illustrates the overview of the proposed framework operating in two primary phases: data preparation and robot-LLM communication. **Data Preparation** is the initial phase providing necessary contextual information to the LLM. It involves organizing and formatting three types of construction-related and robot-related data. 1) Structured construction-related data encompasses data derived from BIM models and construction schedules via APIs. It includes details about conventional building elements (e.g., walls, floors, columns, and windows) as well as non-building objects that interact with robots (e.g., material storage locations, painting locations, and designated no-go zones). The model data is formatted by the software API into machine-readable formats like JSON and XML to enhance the computation processing by an LLM. 2) Unstructured input is typically

 doi:10.20944/preprints202506.1331.v1

provided directly by construction professionals. This information can be provided both before and during robot operations to supplement or override standard task instructions based on the dynamic conditions observed at the construction site. This input may include specific work instructions and strategies to detect and mitigate potential safety hazards in particular situations. Given that these inputs often originate from professionals in the construction site, they should be in natural human language to facilitate immediate comprehension by LLMs without the need for reformatting. 3) Robot-related information includes both structured and unstructured information about robots and their operations. Properties of a robot and available skills with required input arguments can be written in structured formats. On the other hand, task specifications can be written either in natural language or computer-readable formats. However, the flexibility to use natural language to write task specifications can greatly benefit construction professionals allowing them to model required robot behaviors without deep technical knowledge. Following data preparation, in **robot-LLM communication** phase, the LLM is provide with the contextual information and utilizes its extensive world knowledge to interpret the situation and directly guide robot operations through continuous dialogue with a robot. When a robot performs an action, it submits a prompt describing the initiation, progress, and completion. The LLM processes the prompt from the robot and responds in two formats. It provides instructions in natural language for the human supervisor and in JSON format for the robot. The JSON response is parsed to identify the robot skill to be triggered and the input arguments required for that skill. By pre-informing the LLM with necessary background information, the LLM can generate contextually appropriate instructions that accurately reflect the robot's capabilities and parameters of its skills. Also, complex reasoning required for robot operations is significantly reduced to a simple prompt-response implementation on the robot software side by relying on the LLM with extensive knowledge for the adaptive situational understanding and decision-making.

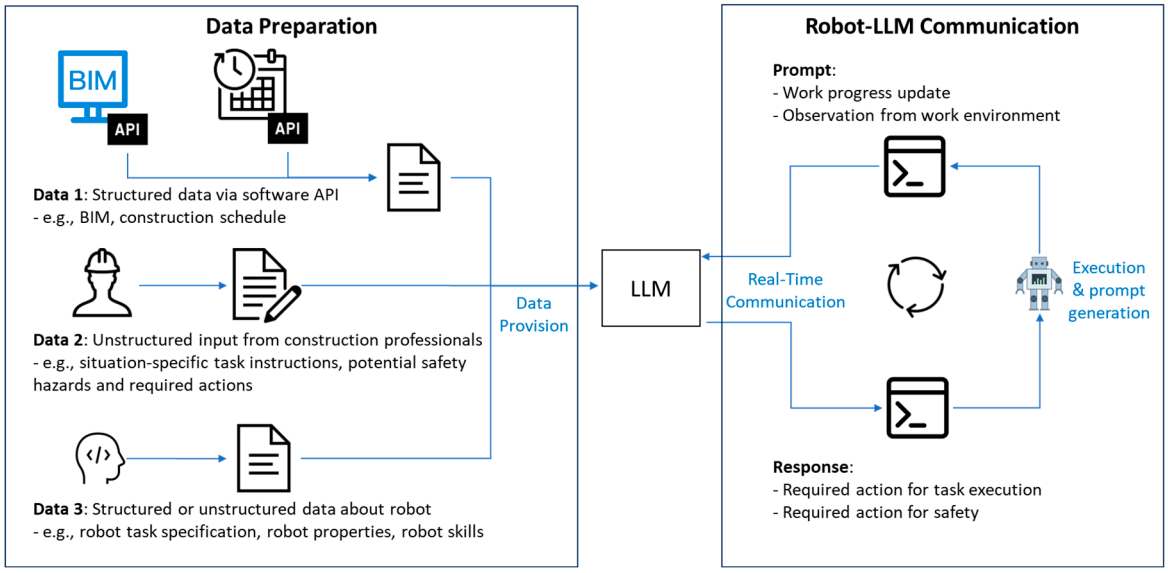

Figure 1. Overview of LLM-based data integration and task planning.

4.2. Painting Robot Software Prototype with BIM-Robot-ChatGPT4 Integration

For the implementation of the framework, this section describes the development of a prototype software program that incorporates BIM, ROS, and ChatGPT4 focusing on enabling the autonomous painting robot operations in simulated construction environments. Figure 2 illustrates the software architecture for the prototype system.

 doi:10.20944/preprints202506.1331.v1

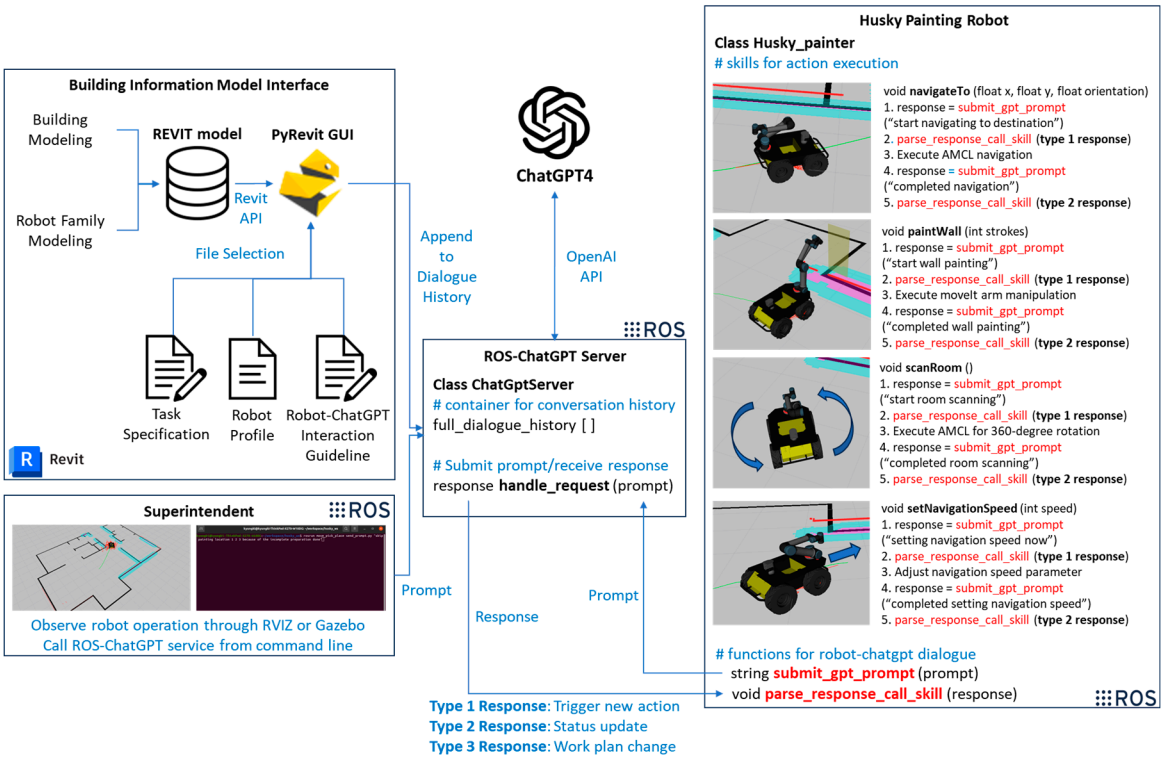

Figure 2. Software architecture for prototype system.

4.2.1. Contextual Data Preparation in BIM Interface

We developed a BIM interface using Autodesk Revit 2022 and the PyRevit API [64] to facilitate the preparation of contextual data. This interface extracts information about building and non-building elements and allows users to select and manage other crucial inputs to be submitted to ChatGPT4, such as robot task specifications, robot profiles, and guidelines for interactions between the robot and ChatGPT. The interface supports the following functions:

- **Extraction of building element data**: Utilizing PyRevit, the interface extracts conventional building element data and formats the data into JSON files. Our approach focuses on extracting only the relevant building element data needed for the painting robot's operations as defined in the human language task specification (see Figure 6 for the painting task specification). Consequently, the scope of the building data extraction in this study is limited to properties of room elements, such as room names and center coordinates, that are mentioned in the painting task specification. The extracted data can vary significantly depending on how the task specifications are written.
- **Modeling of objects related to robot task performance**: Robot task planning requires explicit information that is not typically required by human workers. For instance, the task specification for a painting robot includes actions like "navigate to the painting location" requiring the precise determination of painting locations within the building model. Therefore, this study created custom Revit family models specifically to define painting locations. This approach allows construction superintendents to directly provide precise data for the robot's painting operations by modeling these robot-related objects in the Revit building model. Then, PyRevit is utilized to extract and format this data into JSON files. Like building elements, the family models to be created depend on the type of robot task and how the task specification is written reflecting the unique requirements. For example, for a drywall installation robot, the task specification can state "pick up drywall board from the nearest material storage" which requires the creation of a family model for material storage locations.
- **Creation of robot task specifications**: The interface enables users to select detailed specifications for robot tasks written in natural human language. These specifications describe sequences of

 doi:10.20944/preprints202506.1331.v1

required actions (such as navigating to a room, scanning the room, moving to painting locations) and the robot skills required to perform these actions. The utilization of human language for task specification allows users to describe operational details without in-depth technical knowledge of robotics. Nonetheless, a basic understanding of the robot's capabilities and the appropriate alignment of robot skills with required actions is still required. For instance, if the robot does not have the skill to navigate to a destination, the specification should not include such an action.

- **Configuration of robot profiles**: Through the interface, users can configure robot profiles that specify the robot's properties and available skills. For example, the profile for the "Husky_painter" includes skills, such as navigation, speed setting, painting, and scanning. This profile helps to inform ChatGPT4 of the robot's capabilities facilitating the generation of only executable instructions by ChatGPT4that align with the robot's actual functions.
- **Defining Robot-ChatGPT interaction guidelines**: The interface also allows users to create guidelines for the interaction between the robot and ChatGPT4. These guidelines describe ChatGPT's expected roles, types of responses to be generated, and prompt-response example sets. As shown in Appendix 2, this study makes ChatGPT4 to produce responses in both natural language for the human supervisor and in JSON format for the painting robot. We also defined Type 1 response to trigger new actions, Type 2 response to confirm the current status of action, and Type 3 response to change work plan based on superintendent input.

The BIM interface, operating within a Windows environment, exports all necessary data into a shared Dropbox folder that the Linux-based robot software program can access. Upon initiating operations, the robot software prepends this contextual data to the first prompt it sends to ChatGPT4. The enriched prompt from the BIM interface is then communicated to ChatGPT4 through the ROS-ChatGPT server developed in this study. This data transfer allows the LLM to be equipped with all the necessary background information to generate context-aware responses to instruct the robot's activities.

4.2.2. ROS-ChatGPT Server

This study developed the ROS-ChatGPT server to facilitate synchronous communication between the construction robot, human users, and ChatGPT4. Implemented as a ROS service, this server enables continuous dialogue between the robot's software nodes and ChatGPT4. As the robot executes tasks, it submits prompts describing the initiation and completion of actions. ChatGPT4 responds with JSON-formatted instructions that specify subsequent steps to complete the tasks. These instructions are then parsed by the robot's control software to trigger the appropriate skills. Through this dynamic interaction, the painting robot can continuously receive contextually appropriate instructions from ChatGPT4. Additionally, the ROS-ChatGPT server supports direct input from the construction superintendent who can use either a terminal or a Graphical User Interface (GUI) to submit commands like "skip painting location 1" or "do not paint at locations within 2 meters from human workers." These inputs can be provided both before and during the task execution to modify the task plan or operational parameters based on situational needs. In this simulation-focused study, users can observe robot operations through either the Gazebo simulator, which mimics real-world conditions, or Rviz, which illustrates the robot's internal status, while directly entering commands via the terminal.

The "ChatGptServer" class within this server manages the conversation history to help ChatGPT4 retain contextual data and previous interactions with the robot which is essential for maintaining a coherent dialogue flow. This mechanism allows the LLM to generate responses informed by earlier communications. Without this feature, instructions can lack continuity or relevance to the ongoing tasks. The server processes requests through the "handle_request" method, which submits prompts to ChatGPT4 using OpenAI's "chat.completions.create" API [65]. This API sends conversational prompts that incorporate both the current request and the accumulated dialogue history, enabling the AI to provide responses that are pertinent. This server can be used by

 doi:10.20944/preprints202506.1331.v1

the robot and the human superintendent to interact with ChatGPT4 by submitting prompts and receiving guidance in both natural language and a structured JSON format.

4.2.3. Husky Painter Skills

In the prototype software for the painting robot, "Husky_painter" is the primary class equipping the Husky painting robot with functionalities to connect to the LLM via a ROS service and to execute its skills. The class defines skills for navigating to a location, adjusting navigation speed, performing wall painting, and scanning the room. Each skill initiates with a prompt to the LLM, confirming task start and completion, ensuring seamless integration of task planning and execution. The skills implemented in this study are outlined below:

- navigateTo (float x, float y, float orientation): The "navigateTo" skill directs the robot to move to a specified location with a given orientation. The sequence begins with the robot submitting a prompt to the LLM, requesting permission to initiate navigation with the message "start navigating to destination." Upon receiving a type 1 confirmation response, the robot executes navigation using the Adaptive Monte Carlo Localization (AMCL) [66], which provides accurate location tracking and orientation in the environment. Once the robot reaches the destination, it submits another prompt, "completed navigation," to indicate task completion, and awaits a type 2 response from the LLM, confirming successful execution.
- paintWall (int strokes): The "paintWall" skill enables the robot to perform wall painting actions. The task starts with a prompt to the LLM, "start wall painting," requesting confirmation to proceed. Once a type 1 response is received, the robot utilizes the MoveIt library [67] to control its UR5 arm, performing the specified number of paint strokes at the target location. After completing the painting task, the robot sends another prompt, "completed wall painting," to the LLM to signal completion, and awaits a type 2 confirmation response, indicating successful execution.
- scanRoom (): The "scanRoom" skill allows the robot to perform a 360-degree scan of the room to gather environmental data. The process begins by sending the prompt "start room scanning" to the LLM, which, upon receiving a type 1 response, authorizes the robot to proceed. The robot then initiates a 360-degree rotation using AMCL to scan its surroundings. After completing the scan, it submits a prompt, "completed room scanning," and waits for a type 2 response from the LLM to confirm task completion.
- setNavigationSpeed (int speed): This skill adjusts the robot's navigation speed according to situational needs. The robot starts by sending a prompt, "setting navigation speed now," to the LLM. Upon receiving a type 1 confirmation, it adjusts the internal speed parameter for navigation. After completing the speed adjustment, the robot sends a final prompt, "completed setting navigation speed," to confirm with the LLM, receiving a type 2 response to indicate successful completion.

4.2.4. Skill Execution with Recursive Robot-ChatGPT4 Communication

The implementation of these skills relies on recursive communication with ChatGPT4, utilizing the "submit_gpt_prompt" and "parse_response_call_skill" functions within the robot's control class. These functions are invoked at the initiation and completion of each action to continuously update ChatGPT4 on the robot's status and receive relevant instructions. For example, when initiating navigation to a painting location, the "submit_gpt_prompt" function uses the "handle_request" service within the ROS-ChatGPT server to send the prompt "I started navigating to the destination." According to the task specification, ChatGPT4 may respond with a Type 2 response, which simply confirms the robot's ongoing status without requiring additional actions. Once the robot reaches its destination, it reports back with "I arrived at the destination." At this point, ChatGPT4 sends a Type 1 response, instructing the robot to proceed with the next task – such as painting – according to the task specification. When the robot receives a structured JSON response, like response example 1 in Appendix 2, it first parses the JSON input by identifying the "function_called" field as "navigateToLocation". It then extracts the three arguments – x, y, and orientation – along with their

 doi:10.20944/preprints202506.1331.v1

values. This parsing process relies on hardcoded mappings for each skill, meaning that the robot's software must have pre-defined logic to interpret specific function names and parameter structures. To ensure smooth skill execution, it is essential that the response examples provided to ChatGPT4 align with the robot's skill execution code. By providing these carefully structured response examples to ChatGPT4, we guide it to generate JSON responses that meet the requirements of the robot's parsing software, ensuring consistency between ChatGPT's instructions and the robot's task execution capabilities.

5. Case Study

5.1. Case Study Setup and Data Preparation

This section presents a case study conducted to evaluate how the "Husky Painter" robot adaptively plans its operations based on diverse contextual information and interactions with a superintendent in a simulated construction environment. The case study particularly focuses on an interior wall painting task within a limited workspace where the robot repetitively performs a sequence of actions as described in the task specification. The robot used in this study is the Husky mobile robot equipped with a 2D laser scanner for autonomous navigation and a UR5 arm for painting. As discussed in the objective and scope, this study does not incorporate sensor-based perception of the environment. Instead, all necessary inputs are provided via prompts and by accessing model data in the Gazebo simulation. This allows this case study to concentrate on assessing the robot's capabilities to adaptively plan its operations without hardcoded logic and only via the dialogue with ChatGPT4. As shown in Figure 4, we created a BIM model of a real two-bedroom apartment using Autodesk Revit 2022 for painting task planning within the living room. Nine painting locations were defined in the Revit model using a custom "painting location" family model which includes manually input properties, such as sequence, stroke time, and area. The sequence parameter orders the painting locations, stroke time specifies the repetition of painting strokes, and the area parameter indicates the room where the painting is performed. Figure 4 shows how parameters values and locations of painting location family instances were extracted into JSON files using the "FamilyDataGeneration" plugin developed with PyRevit. Subsequently, as shown in Figure 5, the "FileSelection" PyRevit plugin allows the selection of various necessary files including those for painting task specification (Appendix 1), robot-ChatGPT interaction guideline (Appendix 2), painting locations (Appendix 3), and the robot profile (Appendix 4). These files were then loaded into a Dropbox folder accessible to the Linux-based robot software.

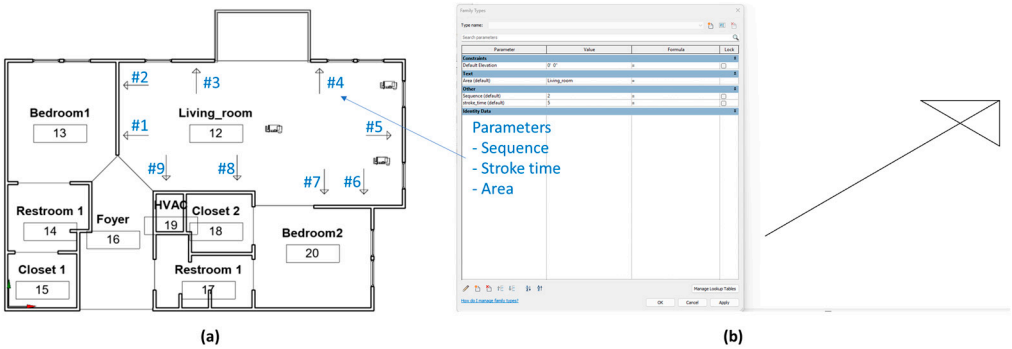

Figure 3. (a) Two-bedroom apartment BIM model, (b) custom family model for painting location.

 doi:10.20944/preprints202506.1331.v1

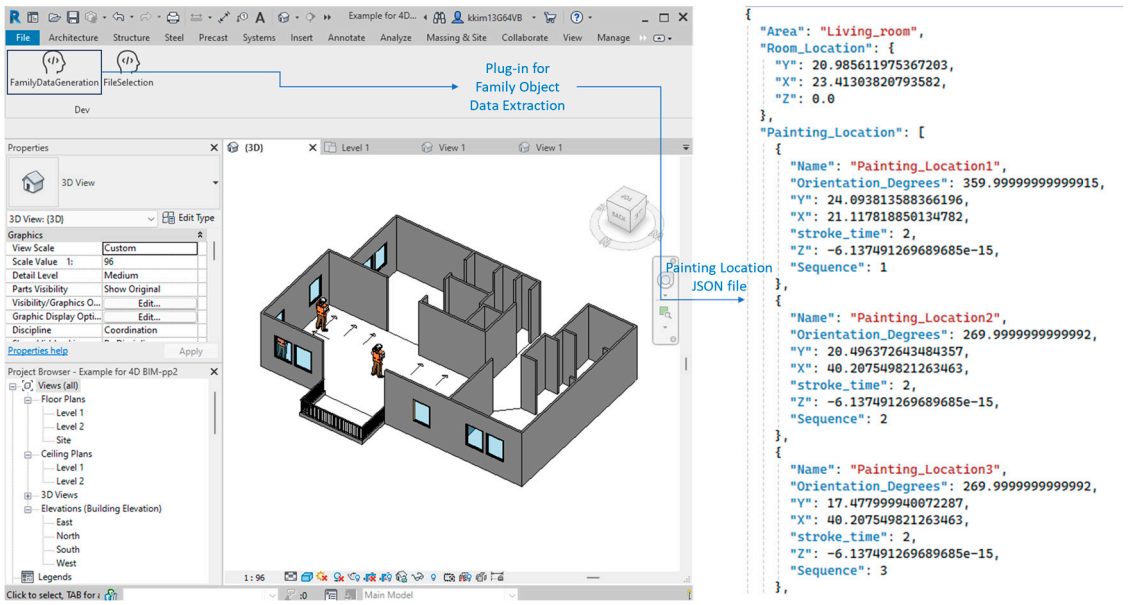

Figure 4. PyRevit GUI for family object data generation.

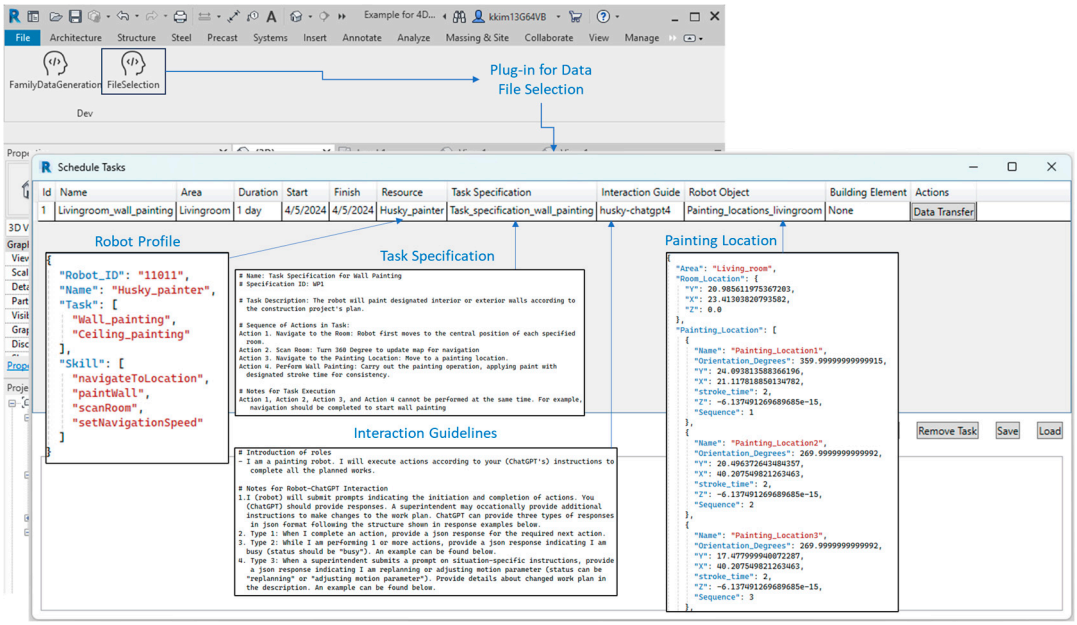

Figure 5. PyRevit GUI for data file selection and data transfer.

Appendix 1 shows the task specification for wall painting that describes the sequence of actions, including navigating to the center of the room, scanning the room to update the map, moving to individual painting locations, and executing wall painting for each painting location. In the skill utilization, each action is associated with the necessary robot skills and input parameters required, such as room center coordinates for navigation and stroke time for painting. Several improvements were made to the task specification based on initial observations of the robot's behavior. For example, actions were explained in separate sentences and explicitly numbered as "Action 1", "Action 2", etc. Also, a note was incorporated to specify that certain actions, like navigation and painting, cannot be performed simultaneously. These adjustments were necessary as they addressed initial errors where the robot attempted to navigate and paint at the same time or executing navigations several times overriding previous commands.

Appendix 2 shows the interaction guideline for communication between the robot and ChatGPT4 that supplements the general task specification. The robot first identifies itself as a painting

 doi:10.20944/preprints202506.1331.v1

robot, tasked with executing actions based on ChatGPT4's directives and additional instructions from a superintendent when necessary. We defined three types of JSON-formatted responses from ChatGPT4 to guide the robot: Type 1 response for initiating the next step upon completion of an action, Type 2 response indicating the robot is busy, and Type 3 response for replanning of the given task or adjustments to the robot motion parameters in response to superintendent inputs. Initially, the lack of specific examples in these guidelines led to errors when the robot attempted to parse incorrectly formatted responses. By incorporating detailed examples into the guidelines, these issues were rectified, and the robot software could correctly execute the instructions.

Finally, a building model creation pipeline was established to convert the Revit 2022 model into a format suitable for Gazebo simulation, involving conversion to FilmBox 3D file (FBX), then to COLLADA file (DAE), and finally incorporating the DAE file into an SDF model for use in Gazebo. A 2D metric map for AMCL navigation [66] was generated using a ROS-based 2D map creator [68] with the Gazebo building model as the input. Figure 6a shows the building model and Hsuky painter spawned in a Gazebo simulation, and Figure 6b shows the metric map and the robot in Rviz software [69] that visualizes the robot's internal status.

The preparation for the case study involved iterative refinements of files in the knowledge base to guide how ChatGPT generated the desired responses, as well as adjustments to the robot's software program to ensure it could accurately parse those responses. During development, several instances of inaccuracies were encountered, such as ChatGPT generating responses that commanded the robot to perform navigation and painting simultaneously, incorrectly formatted JSON outputs, and misinterpretations where the robot assumed painting was required at only one location, leading to navigation without painting at subsequent locations. These issues were addressed through iterative improvements. Once these refinements were finalized, the painting task was consistently guided by ChatGPT4. Although the specific wording of each action occasionally varied, the overall meaning and JSON outputs remained accurate.

The evaluation of the software was conducted in two phases: 1) an initial robot-ChatGPT dialogue to assess basic interaction and task execution capabilities, and 2) an extended robot-ChatGPT dialogue incorporating superintendent input for dynamic work plan adjustments and situation-specific robot control to test the system's responsiveness to changes and unforeseen site conditions. Each phase was repeated 10 times, and in all trials, the tasks were successfully completed, with only slight variations in the phrasing of responses.

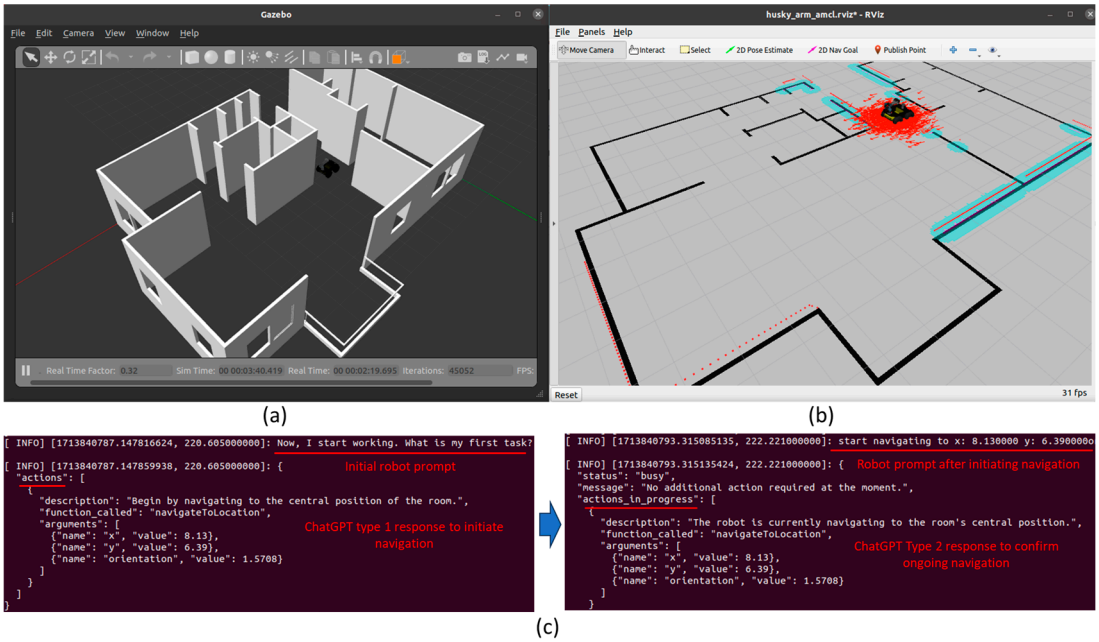

Figure 6. (a) Task initiation in Gazebo simulation, (b) robot internal status in Rviz, and (c) prompts and response.

 doi:10.20944/preprints202506.1331.v1

5.2. Painting Task Planning Via Robot-ChatGPT Dialogue

As shown in Figure 6, the robot initiated the operation by submitting a prompt "Now, I start working. What is my first task" to together with the contextual information. This received a Type 1 response to trigger the first action specified in the task specification. The superintendent received a directive in natural language, "Begin by navigating to the central position of the room" while the robot received a JSON-formatted response instructing it to "navigateToLocation" with input arguments specified in the painting location file. As navigation commenced, the robot submitted another prompt "start navigating to XYZ" generated during skill execution. The robot then received a Type 2 response indicating "actions in progress" which clarified the ongoing action without triggering a new action. The robot software was designed to process responses based on whether the response contained "actions" or "actions in progress" properties. Then, as illustrated in Figure 7, the robot submitted the prompt "completed navigating to XYZ" after completing navigation, and a Type 1 response from ChatGPT triggered the next room scanning action. Following the initiation of room scanning, a Type 2 response described the ongoing action.

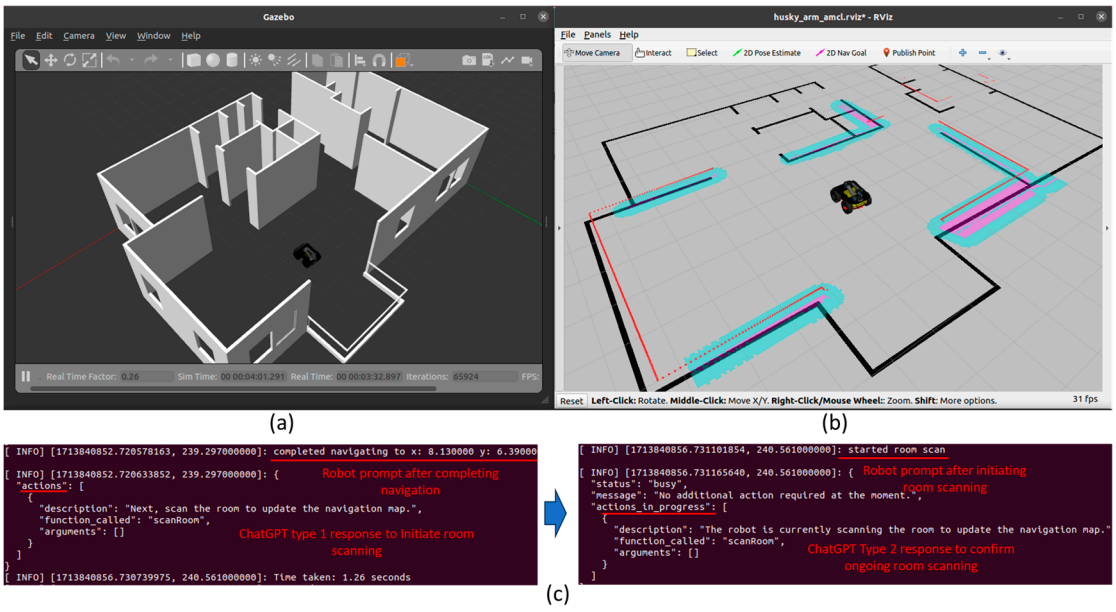

Figure 7. (a) Robot receiving room scanning instruction, (b) robot internal status in Rviz, and (c) prompts and response.

Figure 8 shows the robot navigating to each painting location and executing wall painting as per the specifications in the painting location document. ChatGPT provided sequential instructions like "You've arrived at the fifth painting location. Please start painting the wall" and "You've reached the final painting location. Start painting the wall" communicated in both natural language and JSON format. The operation concluded with the message "The final wall painting task has been successfully completed. Well done!" This case study demonstrates the feasibility of using natural language commands to trigger robot actions based on a predefined task specification. Although the task was initiated by the robot requesting the first action, it could also start with a user prompt such as, "I am the superintendent. Make the robot complete all painting tasks according to the plan." A total of 43 prompt-response exchanges occurred between the robot, user, and ChatGPT to complete painting across all 9 locations, as summarized in Table 1.

Table 1. 43 interactions average response time 4.3209 seconds.

 doi:10.20944/preprints202506.1331.v1

#	Prompt	Response	Response Time (seconds)	#	Prompt	Response	Response Time (seconds)
1	Robot: "Now, I start working. What is my first task?"	Type 1 response to navigate to center coordinates	3.8635	23	Robot: "Completed navigating to location 5"	Type 1 response to start painting at fifth location	3.4591
2	Robot: "Start navigating to room center"	Type 2: "No additional action required."	4.9501	24	Robot: "Painting started"	Type 2: "No additional action required."	4.1301
3	Robot: "Completed navigating to room center"	Type 1 response to initiate room scan	3.5210	25	Robot: "Painting completed at location 5"	Type 1 response to navigate to sixth painting location (x: 12.3, y: 5.2)	5.2085
4	Robot: "Started room scan"	Type 2: "No additional action required."	3.3425	26	Robot: "Start navigating to location 6"	Type 2: "No additional action required."	5.2176
5	Robot: "Completed room scan"	Type 1 response to navigate to first painting location (x: 5.2, y: 5.38)	4.5994	27	Robot: "Completed navigating to location 6"	Type 1 response to start painting at sixth location	3.5528
6	Robot: "Start navigating to location 1"	Type 2: "No additional action required."	4.9233	28	Robot: "Painting started"	Type 2: "No additional action required."	3.7013
7	Robot: "Completed navigating to location 1"	Type 1 response to start painting at first location	3.7831	29	Robot: "Painting completed at location 6"	Type 1 response to navigate to seventh painting location (x: 10.86, y: 5.2)	4.8954

 doi:10.20944/preprints202506.1331.v1

8	Robot: "Painting started"	Type 2: "No additional action required."	4.4186	30	Robot: "Start navigating to location 7"	Type 2: "No additional action required."	4.7253
9	Robot: "Painting completed at location 1"	Type 1 response to navigate to second painting location (x: 5.2, y: 7.3)	4.5604	31	Robot: "Completed navigating to location 7"	Type 1 response to start painting at seventh location	3.8591
10	Robot: "Start navigating to location 2"	Type 2: "No additional action required."	4.9233	32	Robot: "Painting started"	Type 2: "No additional action required."	3.5528
11	Robot: "Completed navigating to location 2"	Type 1 response to start painting at second location	3.7013	33	Robot: "Painting completed at location 7"	Type 1 response to navigate to eighth painting location (x: 7.86, y: 5.2)	5.2176
12	Robot: "Painting started"	Type 2: "No additional action required."	3.7013	34	Robot: "Start navigating to location 8"	Type 2: "No additional action required."	5.6109
13	Robot: "Painting completed at location 2"	Type 1 response to navigate to third painting location (x: 6.46, y: 7.32)	4.5947	35	Robot: "Completed navigating to location 8"	Type 1 response to start painting at eighth location	3.6812
14	Robot: "Start navigating to location 3"	Type 2: "No additional action required."	4.7253	36	Robot: "Painting started"	Type 2: "No additional action required."	4.0332

 doi:10.20944/preprints202506.1331.v1

15	Robot: “Completed navigating to location 3”	Type 1 response to start painting at third location	3.5528	37	Robot: “Painting completed at location 8”	Type 1 response to navigate to ninth painting location (x: 5.42, y: 5.2)	5.3178
16	Robot: “Painting started”	Type 2: “No additional action required.”	3.8591	38	Robot: “Start navigating to location 9”	Type 2: “No additional action required.”	5.3178
17	Robot: “Painting completed at location 3”	Type 1 response to navigate to fourth painting location (x: 10.7, y: 7.32)	5.6109	39	Robot: “Completed navigating to location 9”	Type 1 response to start painting at ninth location	4.0332
18	Robot: “Start navigating to location 4”	Type 2: “No additional action required.”	5.2085	40	Robot: "Painting started at final location"	Type 2: "No additional action required."	2.9016
19	Robot: “Completed navigating to location 4”	Type 1 response to start painting at fourth location	3.0943	41	Robot: "Painting in progress at final location"	Type 2: "No additional action required."	4.0332
20	Robot: “Painting started”	Type 2: “No additional action required.”	4.0332	42	Robot: "Painting completed at final location"	Type 1: "All painting tasks successfully completed."	3.2565
21	Robot: “Painting completed at location 4”	Type 1 response to navigate to fifth painting location (x:	4.4736	43	Superintendent: "Summarize works completed"	Response from ChatGPT: "The robot 'Husky_painter' has successfully completed all the assigned	5.5387

 doi:10.20944/preprints202506.1331.v1

		12.29, y: 5.88)				wall painting tasks..."	
22	Robot: "Start navigating to location 5"	Type 2: "No additional action required."	5.1159				

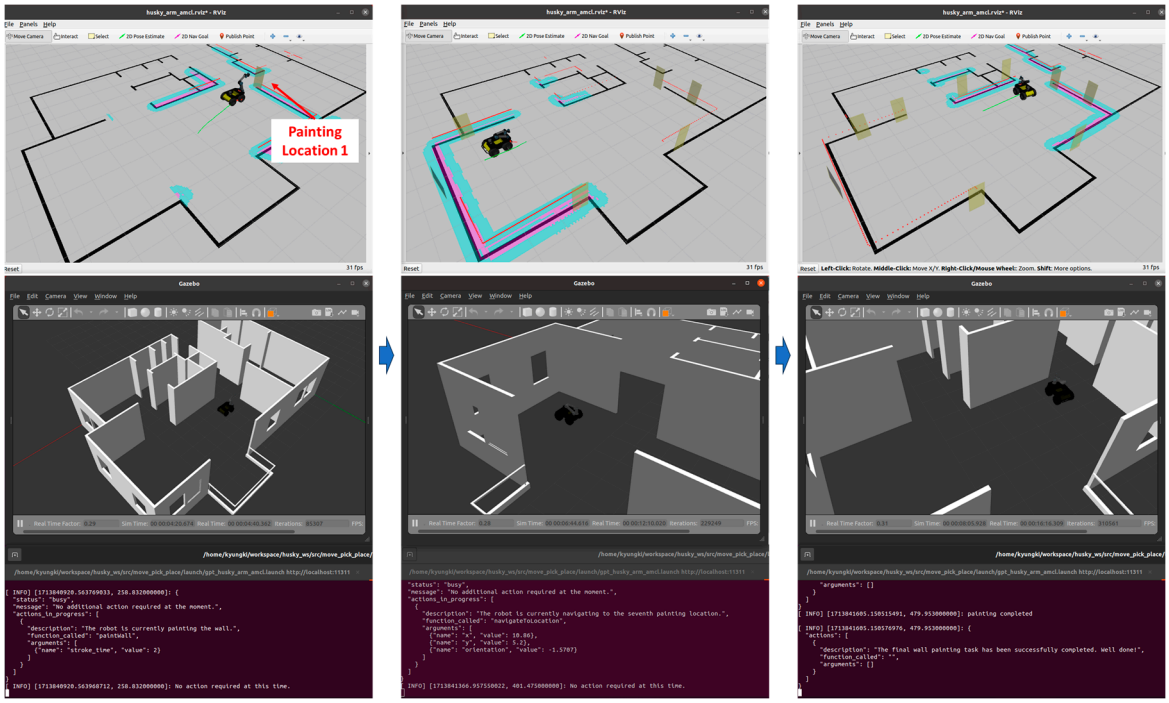

Figure 8. Robot finishing planned wall painting task at nine locations.

5.3. Task Planning Via Robot-ChatGPT Dialogue and Superintendent Input

The second case study involved input from the superintendent who submitted prompts via the command line to dynamically influence the robot's operations. As shown in Figure 9, the first superintendent prompt "skip painting location 1 2 3 because of the incomplete preparation done" led to a Type 3 "replanning" response from ChatGPT4 stating "The robot is now changing its work plan based on the current instructions to skip painting locations 1, 2, and 3." This modification in the robot's task sequence shows the robot's adaptation to real-time human input impacting the operational goals that is essential in construction projects. Then, another prompt from the superintendent "reduce navigation speed to half when you approach painting location 5 (not now) due to many workers there. Then increase the speed when you are done with painting at location 7" was submitted. This resulted in a Type 3 "adjusting motion parameter" response from ChatGPT4 which directed "The robot will reduce its navigation speed to half when approaching painting location 5 due to workers present. After completing painting at location 7, the robot will resume its normal speed." This adjustment shows the robot's adaptation of motion control parameters in a dynamic construction environment based on situational commands.

 doi:10.20944/preprints202506.1331.v1

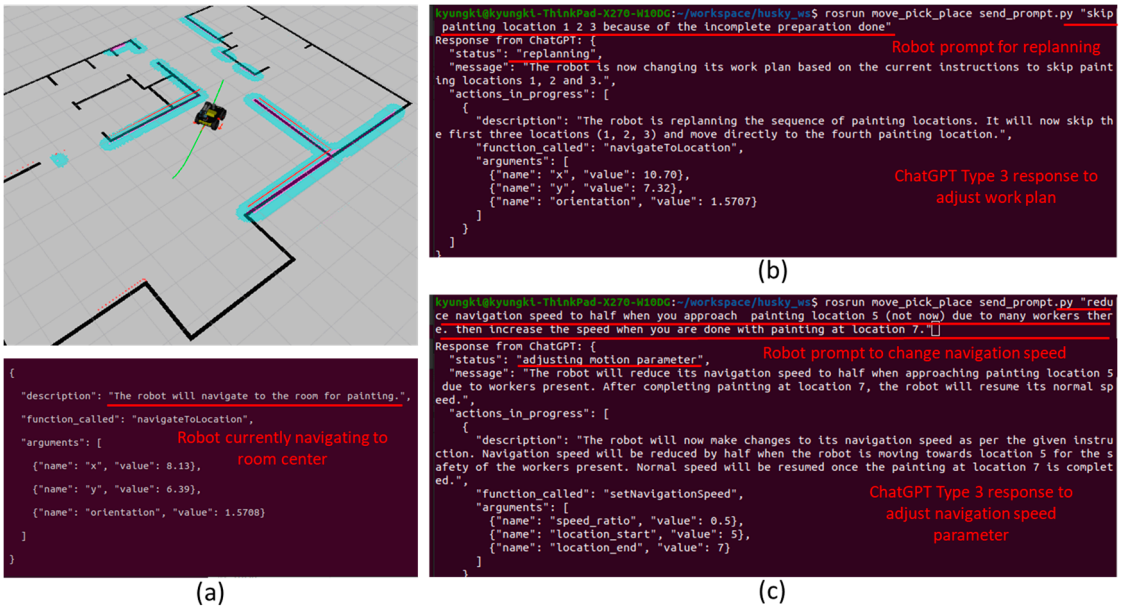

Figure 9. (a) Robot navigating to room center, (b) prompt and response for replanning, (c) prompt and response for motion parameter change.

As depicted in Figure 10, the robot successfully executed these instructions, directly proceeding to painting location 4 and skipping the first three locations. Upon approaching painting location 5, it reduced its navigation speed as instructed and later resumed normal speed after completing the task at location 7. Finally, as shown in Figure 11, the superintendent submitted a concluding prompt to summarize the tasks completed and interactions with the superintendent. This case study highlights the effective integration of superintendent inputs into the robotic operation, allowing for on-the-fly adjustments to the work plan and demonstrating the system's potential for enhanced adaptability in real-world construction scenarios. A total of 32 prompt-response exchanges occurred between the robot, user, and ChatGPT to complete painting while following the instructions from the superintendent, as summarized in Table 2.

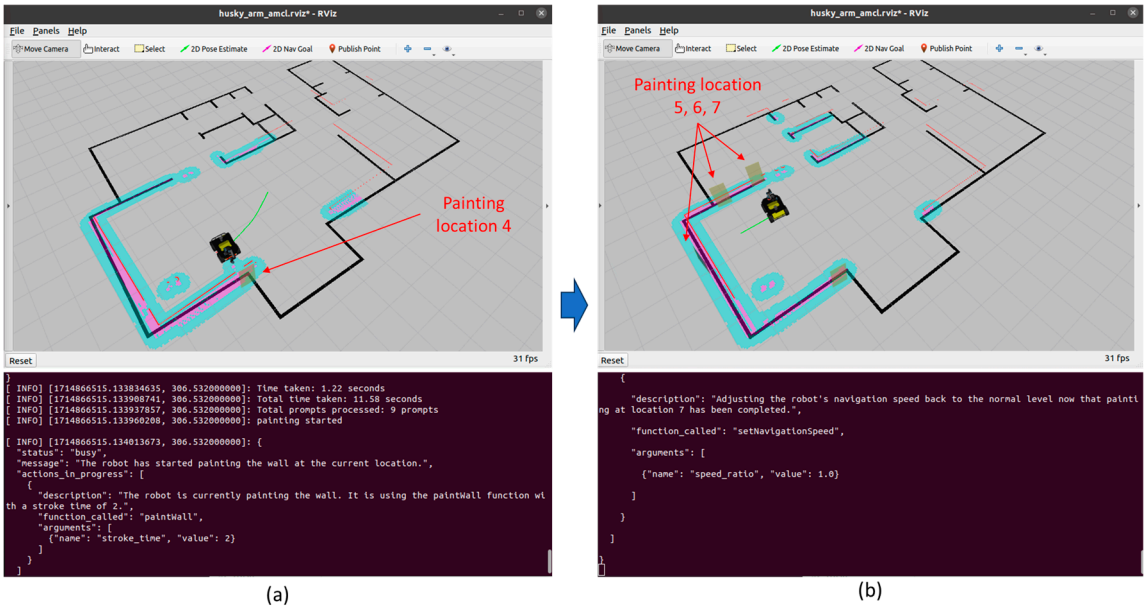

Figure 10. (a) Robot painting at location 5, (b) Robot increasing navigation speed at location 7.

 doi:10.20944/preprints202506.1331.v1

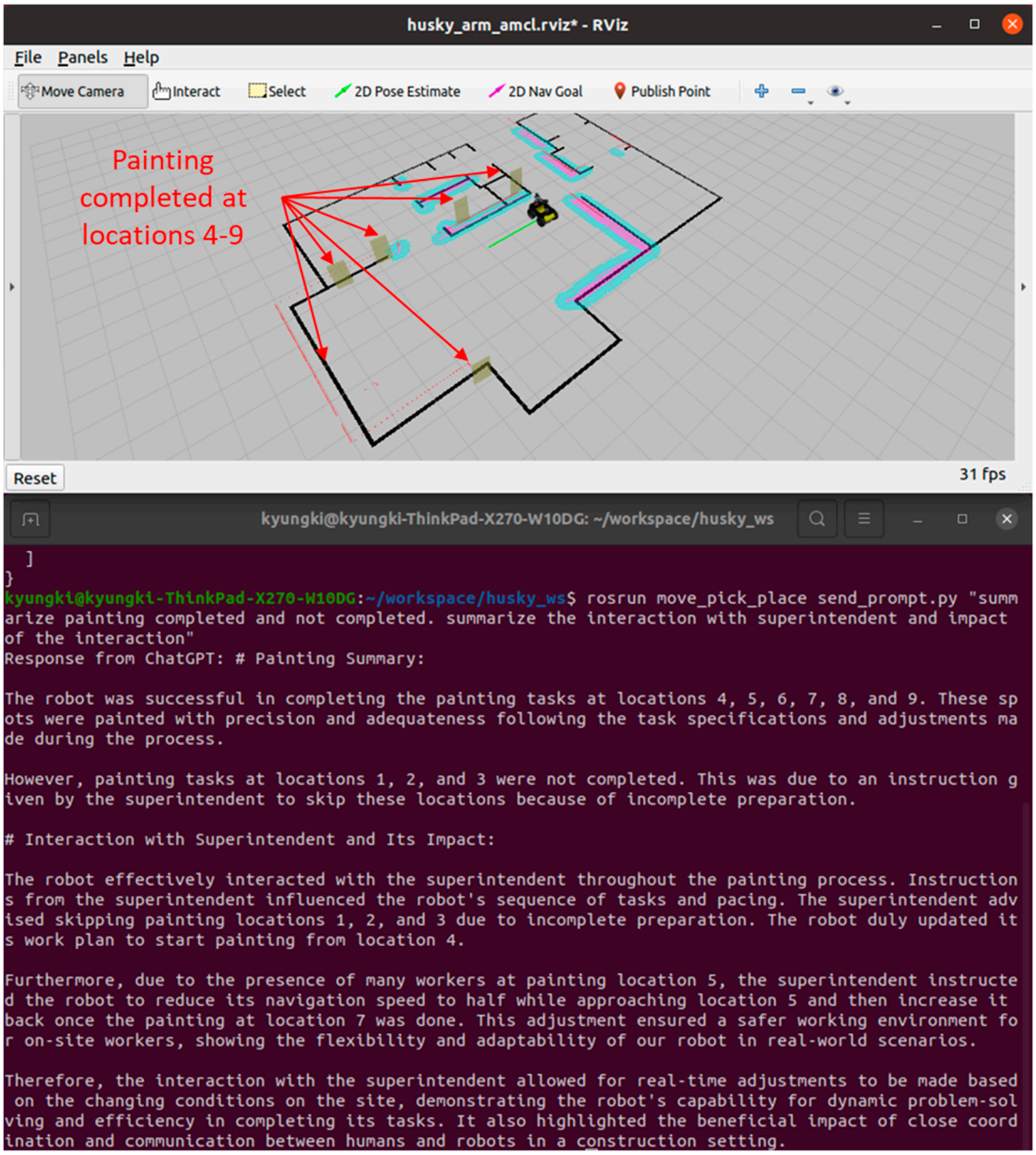

Figure 11. Summarization of task execution and interaction with superintendent.

Table 2. 32 interactions average response time 4.8627 seconds.

#	Prompt	Response	Response Time (seconds)	#	Prompt	Response	Response Time (seconds)
1	Robot: “Now, I start working. What is my first task?”	Type 1 response to navigate to center coordinates	4.4922	17	Robot: “completed navigating to location 6”	Type 1 response to start painting at sixth location	3.7263
2	Robot: “start navigating to room center”	Type 2: “No additional	4.8897	18	Robot: “painting started”	Type 2: “No additional	3.3481

 doi:10.20944/preprints202506.1331.v1

		action required."				action required."	
3	Superintendent: "Skip painting locations 1, 2, 3"	Type 3: "the robot is replanning the sequence.."	6.5843	19	Robot: "painting completed"	Type 1 response to navigate to seventh painting location	5.3984
4	Superintendent: "Reduce navigation speed to half at location 5, increase after location 7"	Type 3 response: "Navigation speed reduced for location 5.."	6.1421	20	Robot: "start navigating to location 7"	Type 2: "No additional action required."	5.0772
5	Robot: "completed navigating to room center"	Type 1 response to initiate room scan.	2.7408	21	Robot: "completed navigating to location 7"	Type 1 response to start painting at seventh location	3.7815
6	Robot: "started room scan"	Type 2: "No additional action required."	3.5857	22	Robot: "painting started"	Type 2: "No additional action required."	3.0984
7	Robot: "completed room scan"	Type 1 response to navigate to fourth painting location	5.1834	23	Robot: "painting completed"	Type 1 response to navigate to eighth painting location & increase navigation speed	6.4008
8	Robot: "start navigating to location 4"	Type 2: "No additional action required."	5.8110	24	Robot: "start navigating to location 8"	Type 2: "No additional action required."	4.5917
9	Robot: "completed navigating to location 4"	Type 1 response to start painting at	3.4877	25	Robot: "completed navigating to location 8"	Type 1 response to start painting at eighth location	4.4586

 doi:10.20944/preprints202506.1331.v1

		fourth location.					
10	Robot: "painting started"	Type 2: "No additional action required."	3.2121	26	Robot: "painting started"	Type 2: "No additional action required."	4.5595
11	Robot: "painting completed"	Type 1 response to navigate to fifth painting location & reduce navigation speed	5.3238	27	Robot: "painting completed"	Type 1 response to navigate to ninth painting location	5.3106
12	Robot: "start navigating to location 5"	Type 2: "No additional action required."	4.5958	28	Robot: "start navigating to location 9"	Type 2: "No additional action required."	4.8117
13	Robot: "completed navigating to location 5"	Type 1 response to start painting at fifth location	3.6059	29	Robot: "completed navigating to location 9"	Type 1 response to start painting at ninth location	3.5661
14	Robot: "painting started"	Type 2: "No additional action required."	4.0348	30	Robot: "painting started"	Type 2: "No additional action required."	3.8647
15	Robot: "painting completed"	Type 1 response to navigate to sixth painting location	5.7541	31	Robot: "painting completed"	Type 2: "Robot completed all the work."	2.9089
16	Robot: "start navigating to location 6"	Type 2: "No additional action required."	4.6873	32	Superintendent: "summarize work and interaction"	Type 3 to summarize works done and interaction with	16.5739

 doi:10.20944/preprints202506.1331.v1

						superintendent	

6. Conclusions and Discussion

This study was initiated to address the complexities involved in processing various formats of information from diverse sources and generating contextually appropriate robot actions for construction robots. Despite the proven capabilities of LLMs, there remains a significant gap in understanding their potential to embed situational awareness and facilitate adaptive task planning into construction robots. As the main contribution, this research created an overarching framework for construction robot task planning that encompasses data preparation phase and robot task planning phase. A prototype software program focusing on an indoor wall painting robot was developed to test whether the framework's integration of robot and LLM could enable the robot to make contextually appropriate decisions. The case study results confirmed that real-time communication between the robot and ChatGPT4 successfully converted structured and unstructured inputs – such as BIM data, natural language instructions, and robot profiles – provided both before and during task execution, into proper instructions for the robot. Also, the use of LLM provided significant advantages over traditional programming methods, which often rely on hardcoding or complex knowledge representation schemes like ontologies that require manual specification of data relationships. By utilizing an LLM as the core for robotic reasoning, the robot control software only needs to maintain a set of defined skills, while the LLM handles the alignment and execution of these skills according to each unique situation. This approach removes the need for numerous if-then statements, which are often rigid and do not adapt well to new or unexpected scenarios. The adaptability provided by the LLM framework allows for more flexible and context-aware task execution. It is important to note that ChatGPT4 was not fully aware of the robot software architecture; it interacted only with exposed functions and input arguments. As a result, users still require a basic understanding of robot software, including the existing functions, their interrelationships, possible combinations of input arguments, and technical constraints on concurrent skill execution (e.g., two navigation-related skills cannot be executed simultaneously, whereas sensing and arm motion might be implemented together). This foundational knowledge is vital for writing practically usable task specifications and interaction guidelines that ensure effective and efficient robot operations. For instance, the three types of responses included in the robot-LLM interaction guideline could be developed considering both the perspectives of software developers and construction professionals.

7. Limitations and Future Directions

Although this study provides a critical framework to integrate LLMs, it also identifies several limitations that should be addressed to achieve greater automation levels. 1) One notable challenge is that, despite the significant reduction in programming requirements, essential data generation and manipulation still rely on hardcoded elements. For instance, the extraction and formatting of building element data from a BIM model requires hardcoded processes tailored to specific task specifications. Additionally, the robot control software is intricately linked to these task specifications, meaning that any modifications to the types of LLM responses or the input arguments required by the robot's skills could impact how the software processes these responses. To overcome these limitations, future studies could expand the capabilities of LLMs to handle data extraction directly from BIM within the framework, aid users in writing task specifications informed by an understanding of the robot software architecture, and automatically generate robot software that reflects requirements outlined in the task specifications. 2) Moreover, the implementation of LLM-robot communication in this study was dependent on the online ChatGPT API, with response times varying based on server status. In both Experiment 1 and Experiment 2, the average response time ranged between 2 seconds

 doi:10.20944/preprints202506.1331.v1

and over 4 seconds for routine actions, while more complex tasks, such as replanning and adjusting motion parameters, typically took over 5 seconds. Such long response times and variability could pose significant risks in safety-critical scenarios in construction robotics. A potential solution could involve training a locally installed LLM to minimize response times, while still leveraging an online LLM for high-level decision-making. Additionally, incorporating minimal if-then statements with LLM-tuned parameters directly into the robot software could enable immediate responses to observed situations. 3) Future research could also explore the application of this framework to other mobile construction robots, conduct multi-robot simulations, and incorporate dynamic models of human worker movements to further optimize task efficiency and safety. Also, exploring the feasibility of allowing concurrent robot actions, such as simultaneously navigating and operating the robot arm, would allow for the development of more efficient robot operations optimizing task sequences and reducing the overall time required to complete construction activities. 4) Building on the framework presented in this study, superintendents working with robots may directly participate in the task planning stage by creating human-language task specifications and specifying work locations or objects for the robot to handle through a BIM-based user interface. For example, the superintendent can generate an hourly or daily basis work plan for the painting robot by inserting painting location objects in a BIM model as demonstrated in the case study. Additionally, superintendents can observe real-time robot operations through robot cameras, or via the robot's internal representation in RVIZ. For practical implementation, both superintendents and workers should have a GUI or an interface for verbal commands to adjust task plans or robot behaviors (e.g., navigation speed, arm movement speed) in real-time. These methods – such as superintendents participating in robot task planning, adding robot-related objects and properties, and writing robot task instructions – are unconventional approaches not commonly employed in current BIM or construction planning workflows without considering robots. However, as the adoption of robots in construction projects increases, such practices may become necessary to ensure seamless integration and optimal utilization of robotic systems. While task-related information in the BIM model is limited to painting locations in this study, the proposed method leverages more comprehensive task-related information provided through other natural language inputs, such as task specifications and LLM-robot interaction guidelines. For example, as demonstrated in the task specification, painting involves executing multiple robot actions, which can be complex to hardcode and challenging to modify in real time. By integrating natural language inputs, the framework enables dynamic and flexible task execution, avoiding the rigidity of hardcoded solutions. Although the scope of the case study focuses on painting tasks, which primarily require painting location objects, the proposed approach is inherently scalable to accommodate more complex construction tasks. For instance, applying the framework to interior wall frame installation could involve additional objects, such as material storage, and a wider range of actions (e.g., navigating to storage, picking up a frame, navigating to the installation location, and placing the frame). Despite these added complexities, the same procedure outlined in the framework can be followed, demonstrating the flexibility and adaptability of the proposed approach to diverse construction scenarios.

Author Contributions: Conceptualization, K.K.; methodology, K.K. and P.-C.H.; software, K.K. and P.-C.H.; validation, K.K. writing—original draft preparation, K.K.; writing—review and editing, P.G.; supervision, K.K.; project administration, K.K.; funding acquisition, K.K.. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

 doi:10.20944/preprints202506.1331.v1

Appendix 1. Robotic Painting Task Specification

Name: Task Specification for Wall Painting
Specification ID: WP1

Task Description: The robot will paint designated interior or exterior walls according to the construction project's plan.

Sequence of Actions in Task:
Action 1. Navigate to the Room: Robot first moves to the central position of each specified room.
Action 2. Scan Room: Turn 360 Degree to update map for navigation
Action 3. Navigate to the Painting Location: Move to a painting location.
Action 4. Perform Wall Painting: Carry out the painting operation, applying paint with designated stroke time for consistency.

Notes for Task Execution
Action 1, Action 2, Action 3, and Action 4 cannot be performed at the same time. For example, navigation should be completed to start wall painting

Skill Utilization:
1. Navigate to the Room: navigateToLocation skill is used with required input of room center coordinate.
2. Scan Room: scanRoom skill is used without any input
3. Navigate to the Painting Location: navigateToLocation skill is used with required input of painting location coordinate.
4. Perform Wall Painting: paintWall is used with required input of stroke time

Appendix 2. Robot-ChatGPT4 Interaction Guideline

Introduction of roles
- I am a painting robot. I will execute actions according to your (ChatGPT's) instructions to complete all the planned works.

Notes for Robot-ChatGPT Interaction
1.I (robot) will submit prompts indicating the initiation and completion of actions. You (ChatGPT) should provide responses. A superintendent may occationally provide additional instructions to make changes to the work plan. ChatGPT can provide three types of responses in json format following the structure shown in response examples below.
2. Type 1: When I complete an action, provide a json response for the required next action.
3. Type 2: While I am performing 1 or more actions, provide a json response indicating I am busy (status should be "busy"). An example can be found below.
4. Type 3: When a superintendent submits a prompt on situation-specific instructions, provide a json response indicating I am replanning or adjusting motion parameter (status can be "replanning" or "adjusting motion parameter"). Provide details about changed work plan in the description. An example can be found below.

Response examples

response example 1 (type 1):

```
{
  "actions": [
    {
      "description": "provide comprehensive description. include contextual details",
      "function_called": "navigateToLocation",
      "arguments": [
        {"name": "x", "value": 3},
```

 doi:10.20944/preprints202506.1331.v1


```
                {"name": "y", "value": 3},
                {"name": "orientation", "value": 3.14}
              ]
            }
          ]
        }

        response example 2 (type 2):
        {
          "status": "busy",
          "message": "No additional action required at the moment.",
          "actions_in_progress": [
            {
              "description": "The robot is currently navigating to the specified location.",
              "function_called": "navigateToLocation",
              "arguments": [
                {"name": "x", "value": 3},
                {"name": "y", "value": 3},
                {"name": "orientation", "value": 3.14}
              ]
            }
          ]
        }

        response example 3 (type 3):
        {
          "status": "replanning",
          "message": "No additional action required at the moment.",
          "actions_in_progress": [
            {
              "description": "Changed work plan based on the input from superintendent.",
              "function_called": "navigateToLocation",
              "arguments": [
                {"name": "x", "value": 3},
                {"name": "y", "value": 3},
                {"name": "orientation", "value": 3.14}
              ]
            }
          ]
}
```


Appendix 3. Painting Locations


```
        {
          "Area": "Living_room",
          "Room_Location": {
              "Orientation_Radians": 1.5708,
              "Y": 6.39,
              "X": 8.13,
              "Z": 0.0
          {
            "Orientation": 1.5707,
            "Y": 7.32,
            "X": 10.70,
            "stroke_time": 2,
            "Z": 0,
            "Sequence": 4
          },
          {
            "Orientation": -1.5707,
            "Y": 5.20,
            "X": 7.86,
            "stroke_time": 2,
            "Z": 0,
            "Sequence": 8
          },
```

 doi:10.20944/preprints202506.1331.v1


```
        },
        "Painting_Location": [
          {
            "Orientation": 3.14,
            "Y": 5.38,
            "X": 5.20,
            "stroke_time": 2,
            "Z": 0,
            "Sequence": 1
          },
          {
            "Orientation": 3.14,
            "Y": 7.30,
            "X": 5.20,
            "stroke_time": 2,
            "Z": 0,
            "Sequence": 2
          },
          {
            "Orientation": 1.5707,
            "Y": 7.32,
            "X": 6.46,
            "stroke_time": 2,
            "Z": 0,
            "Sequence": 3
    },
          {
            "Orientation": 0,
            "Y": 5.88,
            "X": 12.29,
            "stroke_time": 2,
            "Z": 0,
            "Sequence": 5
          },
      {
            "Orientation": -1.5707,
            "Y": 5.2,
            "X": 12.30,
            "stroke_time": 2,
            "Z": 0,
            "Sequence": 6
          },
          {
            "Orientation": -1.5707,
            "Y": 5.2,
            "X": 10.86,
            "stroke_time": 2,
            "Z": 0,
            "Sequence": 7
    },
            {
              "Orientation": -1.5707,
              "Y": 5.20,
              "X": 5.42,
              "stroke_time": 2,
              "Z": 0,
              "Sequence": 9
            }
          ]
}
```


Appendix 4. Robot Profile


```
    {
      "Robot_ID": "11011",
      "Name": "Husky_painter",
      "Task": [
        "Wall_painting",
        "Ceiling_painting"
      ],
      "Skill": [
        "navigateToLocation",
        "paintWall",
        "scanRoom",
        "setNavigationSpeed"
      ]
}
```


References

1. Associated Builders and Contractors. 2024 construction workforce shortage tops half a million. Available online: https://www.abc.org/News-Media/News-Releases/abc-2024-construction-workforce-shortage-tops-half-a-million (accessed on 13 March 2024).
2. CPWR. Fatal and nonfatal injuries in construction. Available online: https://www.cpwr.com/research/data-center/data-dashboards/fatal-and-nonfatal-injuries-in-construction/ (accessed on 13 March 2024).

 doi:10.20944/preprints202506.1331.v1

3. Teicholz, P., Labor-productivity declines in the construction industry: causes and remedies (a second look). AECbytes Viewpoint. 2013.
4. Bock, T., Construction robotics. Auton Robot. 2007, 22, 201–209. doi:10.1007/s10514-006-9008-5.
5. Jones, K., Robots are coming to the construction site. Available online: https://www.constructconnect.com/blog/construction-robotics (accessed on 13 March 2024).
6. Meschini, S.; Iturralde, K.; Linner, T.; Bock, T., Novel applications offered by integration of robotic tools in BIM-based design workflow for automation in construction processes. In Proceedings of the CIB*IAARC W119 CIC 2016 Workshop, Munich, Germany, 2016. Available online: https://mediatum.ub.tum.de/1484218 (accessed on 13 March 2024).
7. Kim, S.; Peavy, M.; Huang, P.; Kim, K., Development of BIM-integrated construction robot task planning and simulation system. Autom Constr. 2021, 127, 103720. doi:10.1016/j.autcon.2021.103720.
8. Correa, F., Robot-oriented design for production in the context of building information modeling. In Proceedings of the International Symposium on Automation and Robotics in Construction, Auburn, AL, USA, 18–21 July 2016. doi:10.22260/ISARC2016/0103.
9. Eastman, C.; Teicholz, P.; Sacks, R.; Liston, K., BIM Handbook, 2nd ed.; John Wiley & Sons: Hoboken, NJ, USA, 2008. doi:10.1002/9780470261309.
10. Sulankivi, K.; Kähkönen, K.; Mäkelä, T.; Kiviniemi, M., 4D-BIM for construction safety planning. In Proceedings of the W099-Special Track 18th CIB World Building Congress, Manchester, UK, 10–13 May 2010; pp. 117–128. Available online: http://cibworld.xs4all.nl/dl/publications/w099_pub357.pdf#page=122 (accessed on 23 April 2016).
11. Koo, B.; Fischer, M., Feasibility study of 4D CAD in commercial construction. J Constr Eng Manag. 2000, 126, 251–260. doi:10.1061/(ASCE)0733-9364(2000)126:4(251).
12. Vaugn, F., 3D & 4D CAD modeling on commercial design-build projects. Comput Civ Eng. 1996.
13. Zhang, S.; Teizer, J.; Lee, J.; Eastman, C.; Venugopal, M., Building information modeling (BIM) and safety: automatic safety checking of construction models and schedules. Autom Constr. 2013, 29, 183–195. doi:10.1016/j.autcon.2012.05.006.
14. Park, J.; Kim, K.; Cho, Y., Framework of automated construction-safety monitoring using cloud-enabled BIM and BLE mobile tracking sensors. J Constr Eng Manag. 2017, 143, 05016019. doi:10.1061/(ASCE)CO.1943-7862.0001223.
15. Kim, K.; Teizer, J., Automatic design and planning of scaffolding systems using building information modeling. Adv Eng Inform. 2014, 28, 66–80. doi:10.1016/j.aei.2013.12.002.
16. Han, K.; Golparvar-Fard, M., Appearance-based material classification for monitoring of operation-level construction progress using 4D BIM and site photologs. Autom Constr. 2015, 53, 44–57. doi:10.1016/j.autcon.2015.02.007.
17. ROS.org. Powering the world's robots. Available online: https://www.ros.org/ (accessed on 28 July 2021).
18. ROS-Industrial. ROS-Industrial. Available online: https://rosindustrial.org/ (accessed on 4 March 2021).
19. Everett, J.; Slocum, A., Automation and robotics opportunities: construction versus manufacturing. J Constr Eng Manag. 1994, 120, 443–452. doi:10.1061/(ASCE)0733-9364(1994)120:2(443).
20. Zhu, A.; Pauwels, P.; Torta, E.; Zhang, H.; De Vries, B., Data linking and interaction between BIM and robotic operating system (ROS) for flexible construction planning. Autom Constr. 2024, 163, 105426. doi:10.1016/j.autcon.2024.105426.
21. Lakin, R.; Kim, K.; Huang, P., ROS-based robot simulation for repetitive labor-intensive construction tasks. In Proceedings of the 18th IEEE International Conference on Industrial Informatics (INDIN 2020), Warwick, UK, 20–23 July 2020.
22. Lundeen, K.; Kamat, V.; Menassa, C.; McGee, W., Autonomous motion planning and task execution in geometrically adaptive robotized construction work. Autom Constr. 2019, 100, 24–45. doi:10.1016/j.autcon.2018.12.020.
23. Follini, C.; Magnago, V.; Freitag, K.; Terzer, M.; Marcher, C.; Riedl, M.; Giusti, A.; Matt, D., BIM-integrated collaborative robotics for application in building construction and maintenance. Robotics. 2020, 10, 2. doi:10.3390/robotics10010002.

 doi:10.20944/preprints202506.1331.v1

24. Ding, L.; Jiang, W.; Zhou, Y.; Zhou, C.; Liu, S., BIM-based task-level planning for robotic brick assembly through image-based 3D modeling. Adv Eng Inform. 2020, 43, 100993. doi:10.1016/j.aei.2019.100993.
25. Wong Chong, O.; Zhang, J.; Voyles, R.; Min, B., BIM-based simulation of construction robotics in the assembly process of wood frames. Autom Constr. 2022, 137, 104194. doi:10.1016/j.autcon.2022.104194.
26. Zhu, A.; Pauwels, P.; De Vries, B., Smart component-oriented method of construction robot coordination for prefabricated housing. Autom Constr. 2021, 129, 103778. doi:10.1016/j.autcon.2021.103778.
27. Oyediran, H.; Turner, W.; Kim, K.; Barrows, M., Integration of 4D BIM and robot task planning: creation and flow of construction-related information for action-level simulation of indoor wall frame installation. arXiv 2024, arXiv:2402.03602. Available online: https://arxiv.org/abs/2402.03602 (accessed on 13 March 2024).
28. OpenAI. GPT-4 technical report. arXiv 2023, arXiv:2303.08774v6. Available online: https://arxiv.org/abs/2303.08774v6 (accessed on 16 April 2024).
29. Huang, W.; Xia, F.; Xiao, T.; Chan, H.; Liang, J.; Florence, P.; Zeng, A.; Tompson, J.; Mordatch, I.; Chebotar, Y., Inner monologue: embodied reasoning through planning with language models. Proc Mach Learn Res. 2023, 205.
30. Sun, H.; Zhuang, Y.; Kong, L.; Dai, B.; Zhang, C., Adaplanner: adaptive planning from feedback with language models. Adv Neural Inf Process Syst. 2024, 36.
31. Liang, J.; Huang, W.; Xia, F.; Xu, P.; Hausman, K.; Ichter, B.; Florence, P.; Zeng, A., Code as policies: language model programs for embodied control. In Proceedings of the IEEE International Conference on Robotics and Automation, London, UK, 29 May–2 June 2023. doi:10.1109/ICRA48891.2023.10160591.
32. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A., ROS: an open-source robot operating system. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009; Volume 3, p. 5.
33. Kim, K.; Peavy, M., BIM-based semantic building world modeling for robot task planning and execution in built environments. Autom Constr. 2022, 138, 104247. doi:10.1016/j.autcon.2022.104247.
34. ROS.org. urdf. Available online: http://wiki.ros.org/urdf (accessed on 28 November 2021).
35. Yin, H.; Lin, Z.; Yeoh, J., Semantic localization on BIM-generated maps using a 3D LiDAR sensor. Autom Constr. 2023, 146, 104641. doi:10.1016/j.autcon.2022.104641.
36. Ghimire, P.; Kim, K.; Acharya, M., Opportunities and challenges of generative AI in construction industry: focusing on adoption of text-based models. Buildings. 2024, 14, 220. doi:10.3390/buildings14010220.
37. Rane, N., Role of ChatGPT and similar generative artificial intelligence (AI) in construction industry. SSRN Electron J. 2023. doi:10.2139/ssrn.4598258.
38. Saka, A.; Taiwo, R.; Saka, N.; Salami, B.; Ajayi, S.; Akande, K.; Kazemi, H., GPT models in construction industry: opportunities, limitations, and a use case validation. Dev Built Environ. 2024, 17, 100300. doi:10.1016/j.dibe.2023.100300.
39. Taiwo, R.; Bello, I.; Abdulai, S.; Yussif, A.; Salami, B.; Saka, A.; Zayed, T., Generative AI in the construction industry: a state-of-the-art analysis. arXiv 2024, arXiv:2402.09939. Available online: https://arxiv.org/abs/2402.09939 (accessed on 13 March 2024).
40. Zheng, J.; Fischer, M., Dynamic prompt-based virtual assistant framework for BIM information search. Autom Constr. 2023, 155, 105067. doi:10.1016/j.autcon.2023.105067.
41. Prieto, S.; Mengiste, E.; García de Soto, B., Investigating the use of ChatGPT for the scheduling of construction projects. Buildings. 2023, 13, 857. doi:10.3390/buildings13040857.
42. Mohamed Hassan, H.; Marengo, E.; Nutt, W., A BERT-based model for question answering on construction incident reports. In Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2022; pp. 216–231. doi:10.1007/978-3-031-08473-7_20.
43. Chang, Y.; Wang, X.; Wang, J.; Wu, Y.; Yang, L.; Zhu, K.; Chen, H.; Yi, X.; Wang, C.; Wang, Y., A survey on evaluation of large language models. ACM Trans Intell Syst Technol. 2024. doi:10.1145/3641289.
44. Zhang, C.; Chen, J.; Li, J.; Peng, Y.; Mao, Z., Large language models for human–robot interaction: a review. Biomim Intell Robot. 2023, 3, 100131. doi:10.1016/j.birob.2023.100131.

 doi:10.20944/preprints202506.1331.v1

45. Ding, Y.; Zhang, X.; Amiri, S.; Cao, N.; Yang, H.; Kaminski, A.; Esselink, C.; Zhang, S., Integrating action knowledge and LLMs for task planning and situation handling in open worlds. Auton Robot. 2023, 47, 729–754. doi:10.1007/s10514-023-10133-5.
46. Micheli, V.; Fleuret, F., Language models are few-shot butlers. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Virtual, 7–11 November 2021; pp. 9312–9323. doi:10.18653/v1/2021.emnlp-main.734.
47. Li, S.; Puig, X.; Paxton, C.; Du, Y.; Wang, C.; Fan, L.; Akyürek, E.; Anandkumar, A.; Andreas, J.; Mordatch, I., Pre-trained language models for interactive decision-making. Adv Neural Inf Process Syst. 2022.
48. Huang, W.; Abbeel, P.; Pathak, D.; Mordatch, I., Language models as zero-shot planners: extracting actionable knowledge for embodied agents. Proc Mach Learn Res. 2022.
49. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A., Language models are few-shot learners. Adv Neural Inf Process Syst. 2020.
50. Dong, Q.; Li, L.; Dai, D.; Zheng, C.; Wu, Z.; Chang, B.; Xu, J.; Sun, X.; Sui, Z., A survey on in-context learning. arXiv 2022, arXiv:2301.00234. Available online: https://arxiv.org/abs/2301.00234 (accessed on 13 March 2024).
51. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y., React: synergizing reasoning and acting in language models. arXiv 2022, arXiv:2210.03629. Available online: https://arxiv.org/abs/2210.03629 (accessed on 13 March 2024).
52. Ding, Y.; Zhang, X.; Paxton, C.; Zhang, S., Task and motion planning with large language models for object rearrangement. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Detroit, MI, USA, 1–5 October 2023. doi:10.1109/IROS55552.2023.10342169.
53. Chen, Y.; Arkin, J.; Zhang, Y.; Roy, N.; Fan, C., Autotamp: autoregressive task and motion planning with LLMs as translators and checkers. arXiv 2023, arXiv:2306.06531. Available online: https://arxiv.org/abs/2306.06531 (accessed on 13 March 2024).
54. Wang, S.; Han, M.; Jiao, Z.; Zhang, Z.; Wu, Y.; Zhu, S.; Liu, H., LLM^3: large language model-based task and motion planning with motion failure reasoning. arXiv 2024, arXiv:2403.11552. Available online: https://arxiv.org/abs/2403.11552 (accessed on 13 March 2024).
55. Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Fu, C.; Gopalakrishnan, K.; Hausman, K., Do as I can, not as I say: grounding language in robotic affordances. Proc Mach Learn Res. 2023, 205.
56. Bhat, V.; Kaypak, A.; Krishnamurthy, P.; Karri, R.; Khorrami, F., Grounding LLMs for robot task planning using closed-loop state feedback. arXiv 2024, arXiv:2402.08546. Available online: https://arxiv.org/abs/2402.08546 (accessed on 13 March 2024).
57. Singh, I.; Blukis, V.; Mousavian, A.; Goyal, A.; Xu, D.; Tremblay, J.; Fox, D.; Thomason, J.; Garg, A., Progprompt: generating situated robot task plans using large language models. In Proceedings of the IEEE International Conference on Robotics and Automation, London, UK, 29 May–2 June 2023. doi:10.1109/ICRA48891.2023.10161317.
58. Song, C.; Wu, J.; Washington, C.; Sadler, B.; Chao, W.; Su, Y., LLM-planner: few-shot grounded planning for embodied agents with large language models. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 2–6 October 2023; pp. 2998–3009.
59. Driess, D.; Xia, F.; Sajjadi, M.; Lynch, C.; Chowdhery, A.; Ichter, B.; Wahid, A.; Tompson, J.; Vuong, Q.; Yu, T., PaLM-E: an embodied multimodal language model. Proc Mach Learn Res. 2023, 205.
60. You, H.; Ye, Y.; Zhou, T.; Zhu, Q.; Du, J., Robot-enabled construction assembly with automated sequence planning based on ChatGPT: RoboGPT. Buildings. 2023, 13, 1772. doi:10.3390/buildings13071772.
61. Luo, H.; Wu, J.; Liu, J.; Antwi-Afari, M., Large language model-based code generation for the control of construction assembly robots: a hierarchical generation approach. Dev Built Environ. 2024, 19, 100488. doi:10.1016/j.dibe.2024.100488.
62. Park, S.; Wang, X.; Menassa, C.; Kamat, V.; Chai, J., Natural language instructions for intuitive human interaction with robotic assistants in field construction work. Autom Constr. 2024, 161, 105345. doi:10.1016/j.autcon.2024.105345.

 doi:10.20944/preprints202506.1331.v1

63. Gazebo. Robot simulation made easy. Available online: http://gazebosim.org/ (accessed on 28 November 2021).
64. Iran-Nejad, E., pyRevit. Available online: https://pyrevitlabs.notion.site/pyRevit-bd907d6292ed4ce997c46e84b6ef67a0 (accessed on 8 May 2024).
65. OpenAI. Text generation models. Available online: https://platform.openai.com/docs/guides/text-generation (accessed on 8 May 2024).
66. ROS.org. amcl - ROS Wiki. Available online: http://wiki.ros.org/amcl (accessed on 28 November 2021).
67. Chitta, S., Moveit!: an introduction. Stud Comput Intell. 2016, 625, 3–27. doi:10.1007/978-3-319-26054-9_1.
68. Yang, H., GitHub - pgm_map_creator: create pgm map from Gazebo world file for ROS localization. Available online: https://github.com/hyfan1116/pgm_map_creator (accessed on 28 November 2021).
69. ROS.org. rviz: package summary. Available online: http://wiki.ros.org/rviz (accessed on 28 November 2021).