Chair of Computational Modeling and Simulation
TUM School of Engineering and Design
Technical University of Munich

Automated Robotic Planning for Corrective Building Maintenance using LLMs and BIM-driven 3D Scene Graphs

Scientific work to obtain the degree

Master of Science (M.Sc.)

at the TUM School of Engineering and Design
of the Technical University of Munich.

Supervised by Prof. Dr.-Ing. André Borrmann
M. Sc. Fiona Collins
M. Sc. Changyu Du
M. Sc. Maikel Brinkhoff
Chair of Computational Modeling and Simulation

Submitted by Nayun Kim (1122233)
Arcisstraße 21
80333 München
e-Mail: ny.kim@tum.de

Submitted on 02. January 2025

Abstract

Building maintenance, particularly corrective maintenance that requires rapid response to unexpected defects, remains a challenging domain for automation. Although Building Information Modeling (BIM) offers rich data for maintenance, its direct application to robotics presents challenges due to different 3D environment representation approaches. This research proposes a framework that bridges this gap by leveraging Large Language Model (LLM) and a novel BIM-driven 3D Scene Graphs (BIM3DSG) for automated robot task planning in corrective building maintenance. The framework consists of five LLM-based modules that handle semantic searching, robot selection, navigation planning, scanning planning, and task planning. To enable efficient integration of building and robotic domains, this research introduces BIM3DSG, which bridges the gap between volumetric BIM data and robotics-oriented 3D Scene Graphs by mapping BIMSegGraphs, room-wise segmented BIM data, to a new extended 3D Scene Graphs. This unified structure effectively enables the direct use of BIM data in robotic applications while addressing LLM token limitations. Evaluation results demonstrate that while the base GPT-4o model performs well on simpler tasks, it struggles with complex spatial reasoning. The more advanced o1-preview model shows superior performance in handling intricate scenarios but faces challenges in optimal scanning position determination. The framework successfully demonstrates the potential for automating corrective maintenance tasks, though some limitations in scanning optimization remain. This research contributes to the field by: (1) developing a comprehensive high-level planning framework for robotic corrective maintenance (2) introducing BIM3DSG as a unified representation bridging BIM and robotics, and (3) a comprehensive evaluation of LLMs' spatial reasoning capabilities in the Architecture, Engineering, and Construction (AEC) domain through integration with a real-world robotics library.

Contents

Abbreviation

AEC	Architecture, Engineering, and Construction
BIM	Building Information Modeling
BIM3DSG	BIM-driven 3D Scene Graphs
FOV	Field of View
GUID	Global Unique IDentifier
IFC	Industry Foundation Classes
LLM	Large Language Model
MEP	Mechanical, Electrical, and Plumbing
PCA	Principal Component Analysis
PDDL	Planning Domain Definition Language
TUM	Technical University of Munich
UAV	Unmanned Aerial Vehicles

Chapter 1

Introduction

1.1 Background and Motivation

BIM has become a standard data format for controlling and managing building information and serves as a core technology for overseeing the entire building lifecycle from design and construction to maintenance. By digitally representing various data such as building geometry, materials, and construction details, BIM provides an adaptable source of information that can be leveraged during building maintenance to facilitate efficient decision-making.

Building Maintenance is broadly divided into two categories: preventive and corrective maintenance (C. Chen & Tang, 2019):

- **Preventive Maintenance:**
  Planned actions based on personal experience to prevent breakdowns. This includes regular inspections and repairs such as painting and covering work to maintain the building's normal performance.
- **Corrective Maintenance:**
  Unplanned urgent repairs or replacements are required when an item fails to perform its function due to delays or failures in preventive maintenance.

Preventive maintenance refers to tasks such as regular inspections, scheduled painting, and periodic equipment replacements. Since these tasks are predictable and planned, extensive research has been conducted on automating preventive maintenance through advanced technologies such as BIM, machine learning, and robotics, for automated documentation, data management, and inspection processes (C. Chen & Tang, 2019; Halder & Afsari, 2023; Valero et al., 2019).

In contrast, corrective maintenance deals with sudden wall cracks, unexpected leaks, or unforeseen equipment failures. Due to the unpredictable nature and urgency of these tasks, research on automating corrective maintenance remains under-explored (Marocco & Garofolo, 2021). Nevertheless, corrective maintenance accounts for approximately 90–97% of building maintenance activities, comprising mostly unplanned service requests (Mo & et al., 2020). While the methods used for preventive maintenance automation are effective in handling predetermined conditions, applying them directly to corrective maintenance is significantly more complex due to unpredictable malfunctions and diverse scenarios.

1.2 Emerging Technologies: LLM and 3D Scene Graphs

1.2.1 LLMs for Corrective Maintenance

Research on building automation has increasingly focused on leveraging robotics for maintenance tasks, with active advancements across various domains. For example, robots have been employed to investigate problem areas in hazardous or hard-to-reach locations (Brunete et al., 2012; Torok et al., 2014), to perform surveillance and inspection of interior spaces and equipment (López et al., 2013), and to automate construction scheduling or quality control tasks (Hamledari et al., 2020). In addition, vertical climbing robots have been developed to maintain and clean the exterior surfaces of high-rise buildings (Moon et al., 2015). Such robotic systems play an important role in automating preventive building maintenance tasks that are difficult or unsafe for human workers and become especially valuable in situations where rapid response is required for unpredictable corrective maintenance.

Corrective maintenance typically begins when users or occupants discover unexpected issues, and each case can demand urgent repairs. Given the urgency and diversity of these tasks, it becomes important to interpret human instructions quickly and plan flexible responses. Consequently, determining how a robot navigates through a building, in what order tasks should be performed, and which tools or materials are necessary demands high-level decision-making, often required for robotic application.

Recent advances in Artificial Intelligence have introduced LLMs capable of understanding and generating human-like text across diverse domains. Motivated by their powerful language comprehension and generation capabilities, researchers have explored LLMs for robot high-level task planning (Brohan et al., 2023; Driess et al., 2023; Rana et al., 2023; Song et al., 2023). However, LLMs are primarily trained on textual data, which inherently limits their ability to interpret or navigate complex 3D environments (B. Chen et al., 2024). Although studies are beginning to investigate spatial reasoning within LLMs, the field remains underexplored, particularly for building maintenance tasks.

1.2.2 Bridging BIM and 3D Scene Graphs for 3D Representation

To address this challenge, integrating the enriched building information provided by BIM is essential. BIM encompasses a building's lifecycle and contains valuable data on geometry, materials, and construction details. However, feeding the entire BIM dataset directly into an LLM is typically impractical due to token limitations. While such detailed information (e.g., material properties, construction specifications) is beneficial throughout the building lifecycle, it often exceeds what is necessary for applications like robotic path planning or rapid corrective maintenance. Instead, robotic operations require concise yet crucial spatial and semantic information, including navigable spatial structures such as rooms, corridors, doors, and objects that robots can interact with, as well as other spatial data

organized and scanned by rooms. This data directly influences robot route planning, task sequencing, and object interaction.

In this regard, 3D Scene Graphs, initially developed by Armeni et al. (2019), have gained attention in robotics and computer vision as a means to represent core spatial and semantic information (Brohan et al., 2023; Cheng et al., 2024; Rana et al., 2023; Zheng et al., 2023). A 3D Scene Graph selects important components (e.g., rooms, objects, assets, scanning positions) and organizes their spatial connections, attributes, and locations in a compact graph format (Armeni et al., 2019). This graph-based representation enables efficient processing of complex spatial data by focusing on the essential information needed for robotic tasks. It helps overcome token-limit issues in LLMs and provides a clear, compact spatial context that is essential for robotic decision-making. However, conventional 3D Scene Graphs have mostly been tailored to household tasks in robotics and computer vision, often covering only small-scale environments such as domestic spaces. As a result, they do not encompass major building elements (e.g., walls, ceilings, floors, windows), making it difficult to map BIM information to 3D scene graphs for applying robotics to tasks in the AEC domain

On the other hand, there has been active research on converting BIM data into graphs to enhance various building-related tasks (Buruzs et al., 2022; Gan, 2022; Khalili & Chua, 2015; Tauscher et al., 2016; Vilgertshofer & Borrmann, 2017; Zhao et al., 2020). The majority of these studies map volumetric building object instances in the BIM model (e.g., IfcWall, IfcDoor) as nodes in the graph (Zhu et al., 2022). However, such volumetric graphs are not well-suited for applications where a robot needs to access details segmented by individual rooms (e.g., surfaces, objects) for direct interaction.

To address this issue, the BIM-Room-Graph module (Claire, 2024), originally developed for matching scanned data with BIM, presents a potential solution. It achieves this by segmenting the volumetric BIM model into a room-wise structure of surfaces, known as BIMSegGraphs (Claire, 2024). This approach provides a solid foundation for extracting the room-wise data required for robotic applications from the BIM model while maintaining a graph structure similar to conventional 3D Scene Graphs. As a result, it holds considerable potential for integration with existing robotic libraries. Nevertheless, there has been limited exploration into how to effectively combine BIMSegGraphs with conventional 3D Scene Graphs to leverage BIM data for building maintenance for direct robotic applications.

In the following section, I discuss how these emerging technologies intersect with the main challenges of corrective maintenance and introduce the research questions.

1.3 Research Questions

As discussed in section 1.1, previous research on automating building maintenance via robotics has predominantly centered on predictable preventive maintenance (Brunete et al., 2012; Hamledari et al., 2020; López et al., 2013; Moon et al., 2015; Torok et al.,

2014). In contrast, corrective maintenance involves unexpected and unpredictable failures, which necessitates a rapid response to a wide range of scenarios using robotic systems.

In a typical corrective maintenance scenario, once a user reports a problem, the robot must plan its route, determine the task sequence, and select the appropriate tools. LLMs offer a promising solution for high-level planning through their natural language comprehension and domain-specific reasoning capabilities. Nonetheless, LLMs are limited in spatial reasoning, and directly feeding comprehensive BIM data to them is impractical due to token limits.

Recent studies have shown that 3D Scene Graphs can be used to extract and summarize essential building structures for robotic or computer vision tasks, thereby reducing token limitations and enhancing spatial awareness (Brohan et al., 2023; Driess et al., 2023; Rana et al., 2023; Song et al., 2023). However, existing 3D Scene Graph implementations typically focus on relatively domestic environments, limiting their utility in AEC domains where crucial architectural elements (e.g., walls, ceilings, floors, facilities) must be represented. Meanwhile, conventional BIM-driven graphs, which adopt a volumetric object-centric approach, do not easily grant access to the room-wise data required for robotic applications. The introduction of BIMSegGraphs Graphs offers a potential solution by segmenting building models in a room-wise manner, but further research is needed to integrate BIMSegGraphs with standard 3D Scene Graphs and extend the outcome to real-world robot implementations.

To address these challenges, the present study identifies three key research problems:

- Most existing robotic automation in the AEC domain has focused on preventive maintenance tasks, partly due to the unpredictable nature of corrective maintenance.
- Conventional 3D Scene Graphs are useful for robotic task planning, but there is no established process for standardizing the representation of key architectural elements (e.g., doors, walls, windows) in the AEC domain.
- BIM-driven graphs are primarily structured around volumetric objects, which limits room-wise robotic interaction. BIMSegGraphs address this by re-configuring BIM data into room-wise segments, but their integration with existing 3D Scene Graphs has not been fully explored.
- There is limited understanding of how to integrate a robot plan for building maintenance with an actual robotic library (e.g., path planning, task sequencing).

To address these gaps, a comprehensive robot planning framework for corrective maintenance is needed. This framework suggests a method to bridge the gap between BIMSegGraphs and conventional 3D Scene Graphs that constructs a method to generate BIM-driven 3D Scene Graphs. Additionally, it supports LLM-based high-level planning and seamlessly interfaces with a robotic library for corrective maintenance tasks. Accordingly, this study explores the following research questions:

RQ1. **Automated Robot Planning for Corrective Maintenance:** How can I design a robot high-level task planning framework that integrates LLMs to effectively manage the unpredictable corrective building maintenance?

RQ2. **BIM-driven 3D Scene Graph (BIM3DSG):** How can I bridge the gap between BIMSegGraphs and conventional 3D Scene Graphs to directly apply BIM data in robotic applications for the AEC domain?

RQ3. **Integration LLM-based Task Planning with Robotic Library for building maintenance:** How can I integrate such an LLM-based robot task planning approach and BIM-driven 3D Scene Graphs into a robotic library for the AEC domain, so that future robotic operations for building maintenance can be potentially executed?

Chapter 2

Literature Review

2.1 Automation in Building Maintenance

Robots have been employed in building maintenance to automate preventive maintenance tasks by investigating problematic areas in hazardous or hard-to-reach locations (Brunete et al., 2012; Torok et al., 2014). Additional applications include surveillance and inspection of interior spaces and equipment (López et al., 2013), as well as automated construction scheduling or quality control tasks (Hamledari et al., 2020).

Among these, Hamledari et al. (2020) proposed a framework for overall automated task planning from data collection with robots to facility monitoring, leveraging key information from 4D BIM models. In their framework, geometric information, inspection targets, and time-stamping details from the 4D BIM model are extracted and subsequently provided to the robot as essential information. This research contributed to effectively automating construction quality inspection using Unmanned Aerial Vehicless (UAVs). However, the rule-based algorithm employed in the planning phase was limited in handling more diverse scenarios and defect cases required in corrective maintenance tasks.

Furthermore, their study highlighted challenges when applying the BIM model's geometric information to robotics. For instance, for a drone to determine its position when inspecting wall construction quality, a custom algorithm had to be developed to segment walls per room from the BIM model. This challenge arises because the geometric data extracted from BIM models is defined in an object-centric (volumetric) manner, making it difficult to directly access the surface information robots need to interact with within a room. Consequently, a custom algorithm was utilized to extract the wall surfaces. This example underlines the necessity of finding methods for alignment between 3D environments derived from BIM and robotic systems in order to achieve integration.

2.2 3D Scene Graphs vs BIM-driven Graphs

To automate building maintenance tasks using robots, it is essential for the robots to accurately understand 3D environments and leverage this information to make informed decisions. In this regard, the AEC domain typically employs BIM as a standard way of representing 3D environments, enriched with data specific to building maintenance. Meanwhile, there has been growing interest in research focused on using LLMs and 3D Scene Graphs to enable robots to understand and interact with large-scale 3D environments

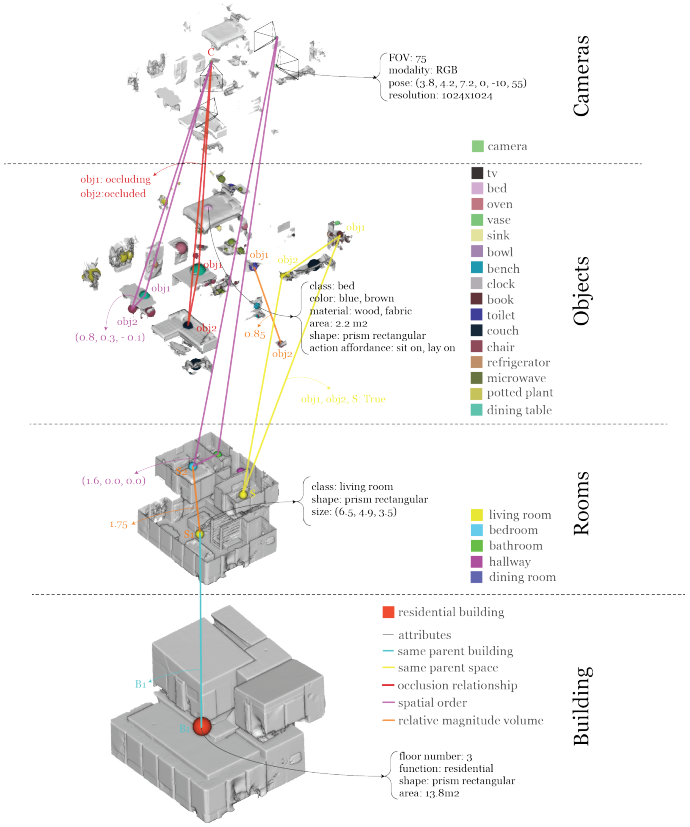


Figure 2.1: 3D Scene Graph Structure from Armeni et al. (2019)

(Cheng et al., 2024; Rana et al., 2023) This section reviews prior studies to examine each approach and discusses the challenges involved in aligning these two representations.

2.2.1 3D Scene Graphs

Various approaches have been explored to convey 3D environment information to robots using LLMs, including vision-based value functions (Brohan et al., 2023), object detectors (Huang et al., 2022; Song et al., 2023), and Planning Domain Definition Language (PDDL) descriptions of a scene, which is a standardized language for defining planning problems, allowing robots to reason about actions and goals (B. Liu et al., 2023; Silver et al., 2022). However, these methods have been constrained to small, limited environments, often only one or two rooms (Rana et al., 2023).

To address the scalability issue to larger spaces, a growing number of LLM-based robot task planning studies have introduced the concept of 3D Scene Graphs (Cheng et al., 2024; Rana et al., 2023). A 3D Scene Graph is a hierarchical graph representation designed to model 3D environments with semantic and spatial information (Armeni et al., 2019). Originally developed for computer vision tasks, they aim to integrate spatial, semantic, and relational information into a unified framework. This structure, widely adopted in robotics,

consists of four layers: (1) the building layer, (2) the room layer, (3) the object layer, and (4) the camera layer, as illustrated in fig. 2.1 (Armeni et al., 2019).

Their hierarchical structure enables semantic searching to extract essential spatial information for LLMs based on user inputs, effectively addressing token limitation issues (Rana et al., 2023). For instance, the SayPlan study utilized 3D Scene Graphs to implement a semantic searching module that reduced token count by 70-80%, enabling the communication of large-scale 3D environments with up to 37 rooms to LLMs. This research also demonstrated the development of a robot task planner using LLMs for household tasks and provided quantitative evaluation metrics.

However, applying such a structure to the AEC domain is challenging. This difficulty stems from the fact that the current 3D scene graph structure does not account for core building elements such as walls, ceilings, floors, windows, and doors. Moreover, as discussed in section 2.1, BIM models do not organize data room-by-room. This makes it challenging to generate 3D scene graphs, which rely on room-specific data for robot interaction, directly from BIM data in the AEC context. This is because most 3D scene graph generation methods rely on raw scanning data from established 3D scanning benchmarks e.g., Matterport3D (Chang et al., 2017), Gibson Database (Xia et al., 2018) (Bae et al., 2023). As a result, 3D Scene Graphs are often structured around camera and room layers (Armeni et al., 2019).

2.2.2 BIM-driven Graphs

In parallel with 3D Scene Graph approaches, numerous studies in the AEC domain have attempted to convert BIM data into graph-based formats to improve efficiency(Buruzs et al., 2022; Gan, 2022; Khalili & Chua, 2015; Tauscher et al., 2016; Zhu et al., 2022). These efforts focus on enhancing building information representation, clarifying relationships between objects, facilitating data integration, and promoting interoperability and efficient query processing (Zhu et al., 2022). For example, Khalili and Chua (2015) converted Industry Foundation Classes (IFC) data into labeled graphs to enable topological queries regarding building elements. Similarly, Tauscher et al. (2016) transformed the IFC object model into a graph to establish a graph-based BIM querying approach. In another study, Gan (2022) utilized the graph database Neo4j to integrate sensor data with IFC information for bridge structural health monitoring. To enrich building space information, Buruzs et al. (2022) developed accessibility graphs from IFC data as an intermediate step.

In most of these approaches, volumetric BIM building objects are mapped to nodes in the graph (Zhu et al., 2022). However, this volume-based mapping often fails to provide the surface-level information inside a room that a robot needs for interactive tasks, thus posing a challenge for direct applicability to robotic applications, as demonstrated in the experiments by Hamledari et al. (2020).

BIM-Room-Graph module (Claire, 2024) introduced a concept called BIMSegGraphs to tackle the challenges stemming from the volumetric nature of standard BIM objects.

By segmenting and merging BIM surfaces on a per-room basis, the BIM-Room-Graph module aligns scanning data with BIM information in a manner that reflects the room-wise data collection approach commonly used in scanning systems. This alignment is further encapsulated in a graph structure that captures adjacency relationships among the resulting room-wise elements and surface segments.

Although BIMSegGraphs provide a key foundation for converting volumetric BIM data into a room-wise data structure that is easier for robotic applications to use, they still lack some of the detailed hierarchical layers found in 3D Scene Graphs. Additionally, they include highly detailed and scanning domain-specific information, not all of which directly correlates with existing robotics libraries, underscoring the need for further investigation into robust methods for integrating BIM data within robotic workflows.

2.3 Large Language Models

Recent advancements in Artificial Intelligence have demonstrated that LLMs possess remarkable capabilities in understanding domain knowledge and interpreting various forms of human natural language without specific training. In this section, I present LLMs research to show its amazing milestones in robotic task planning and explain why I hypothesize that LLMs could potentially address the complexity of corrective maintenance tasks involving diverse cases and variables.

2.3.1 Robot Task Planning using LLMs

As discussed in section 2.1, rule-based algorithms for automated maintenance tasks often struggle to handle the diverse defect cases and unpredictable scenarios found in real-world corrective maintenance. To address these limitations, recent research has begun exploring the potential of large language models (LLMs) for robot task planning (Cheng et al., 2024; Han et al., 2024; Kannan et al., 2024; Rana et al., 2023; Song et al., 2023). These studies leverage LLMs' remarkable capabilities in understanding domain knowledge and interpreting various forms of human natural language without requiring extensive task-specific training. By doing so, they have achieved significant milestones in automating high-level task planning directly from natural language instructions provided by users.

SMART-LLM (Kannan et al., 2024) introduces a framework specifically designed for multi-robot task planning that decomposes high-level instructions into subtasks, forms robot coalitions, and performs task allocation using LLMs. This approach emphasizes coordinating multiple heterogeneous robots by leveraging LLMs' ability to understand task dependencies and robot capabilities.

LLM-Planner (Song et al., 2023) focuses on few-shot grounded planning for single robots in embodied environments. Its key innovation lies in dynamic replanning capabilities, when the robot encounters obstacles or changes in the environment, the system can update its

plans based on real-time observations. This allows for more robust execution in partially observable environments while requiring minimal training data.

LLM-Personalize (Han et al., 2024) addresses the challenge of aligning robot behaviors with individual user preferences. The framework combines imitation learning and iterative self-training to personalize an LLM planner according to specific user needs. This enables household robots to learn and adapt their task execution strategies based on different users' preferences for object arrangements and task completion styles.

These approaches demonstrate how LLMs can be leveraged in different ways to create more flexible and adaptable robot planning systems compared to traditional rule-based methods using their strong ability to interpret human natural language and general knowledge to make high-level decisions. However, these studies revealed limitations of LLMs including environment scalability constraints due to token limits and potential hallucinations stemming from limited spatial reasoning capabilities. In particular, to address the scalability issue of being confined to small 1-2 room environments, 3D Scene Graphs have been proposed as an effective method for conveying 3D environmental information to robots (Rana et al., 2023).

2.3.2 Robot Task Planning using LLMs and 3D Scene Graphs

The SayPlan (Rana et al., 2023) study presents an LLM-based planner that leverages large-scale spatial data by integrating the reasoning abilities of LLM and the hierarchical data management characteristics of 3D scene graphs. In this research, LLM are employed for two primary tasks: semantic searching and task planning. The semantic searching component interprets natural language user input and identifies corresponding target nodes in the 3D scene graph. Initially, in the semantic searching module, the LLM input is restricted to collapsed 3D scene graphs containing only high-level node information. Then it can then "expand" or "close" these nodes through function calling, enabling it to locate the desired target node. This approach reduces input token counts by approximately 70–80%, which in turn allows the LLM to handle information about more than 37 rooms, as opposed to just one or two (Rana et al., 2023).

Once the target node is found, SayPlan (Rana et al., 2023) employs an LLM-based planner to devise high-level plans for the given task. Subsequently, low-level task planning, such as generating final robot action commands or navigation paths, is handled by algorithms like Dijkstra's algorithm. This study is the first to introduce a semantic searching module utilizing 3D scene graphs, enabling the LLM to identify target nodes through its own spatial reasoning capabilities. This advancement demonstrates the scalability of 3D environment data for LLM task planning, expanding its applicability from one or two rooms to environments encompassing up to 37 rooms (Rana et al., 2023).

However, since the LLM's inherent spatial reasoning does not always immediately identify the target node, the authors developed a scene graph simulator to simulate and provide feedback on search results, allowing the LLM to regenerate the plan. This suggests that

LLMs, particularly GPT-4 and GPT-3.5 models, have certain limitations in their own spatial reasoning capabilities, although this study does not identify specific constraints regarding this limitation. Furthermore, while this study focuses on semantic spatial reasoning (i.e., understanding user instructions and identifying corresponding nodes), it does not extensively explore spatial dimensional reasoning, such as leveraging precise coordinate or dimension data for more complicated spatial tasks.

2.3.3 Robot Path Planning using LLMs

Robot path planning has traditionally relied on classic graph searching algorithms such as Dijkstra's and A* due to their guaranteed optimality, implementation simplicity, reliability, and computational efficiency (L. Liu et al., 2023). However, recent research has begun exploring the potential of LLMs in path planning tasks (Cheng et al., 2024; Meng et al., 2024).

Meng et al. (2024) demonstrated this potential through LLM-A*, which combines LLMs with the A* algorithm. By leveraging LLMs to generate waypoints and guide the A* search process, their hybrid approach achieved significant reductions in both computational operations and memory usage compared to traditional A*, while maintaining path validity.

While LLMs have not yet shown sufficient stability to fully replace classical algorithms in path planning, due to limited research on their spatial reasoning capabilities, they offer a crucial advantage on semantic understanding. Unlike traditional algorithms that focus solely on geometric path planning, LLMs combined with 3D scene graphs can provide richer semantic comprehension of the environment (Cheng et al., 2024).

Cheng et al. (2024) propose and evaluate an LLM-based navigation path planning framework integrated with 3D scene graphs, leveraging the concept of LLM Tooling. This study contributes by assessing LLMs' ability to interpret 3D scene graphs, identifying optimal prompting strategies, and employing LLM-based scoring mechanisms to enhance the accuracy and reliability of LLM responses. Specifically, the study leverages the notion of LLM Tooling, in which an LLM is used not only to produce answers but also to evaluate those answers and provide feedback, thereby iteratively improving navigation path planning accuracy. Moreover, the authors experiment with various 3D scene graph representations, including hierarchical versus flattened 3D scene graphs and those with semantic node descriptions versus those without, to identify the most suitable representation for LLM-based navigation path planning.

However, the evaluation in this study is limited. With only five trial attempts and a relatively simple evaluation procedure, it remains challenging to fully understand the limitations of LLMs' spatial reasoning capabilities and to conclusively determine the optimal 3D scene graph structure for LLM-based navigation planning.

2.4 Summary of Literature Review

Based on the above literature review, the current research in automation in building maintenance, robotics with LLMs and 3D scene graphs can be summarized as follows.

Current research in building maintenance automation primarily focuses on preventive maintenance tasks, leveraging their predictability and the possibility of planning ahead. However, this approach has limitations in addressing corrective maintenance tasks, which require rapid adaptation to unforeseen conditions and dynamic data. When it comes to representing 3D environments for robotic applications, a mismatch exists between conventional 3D scene graphs, which focus on rooms and objects but omit fundamental building elements like walls, ceilings, and floor, and BIM-derived data, which emphasize volumetric structural information but often lack room-wise segmented data that robots need to interact. Although BIMSegGraphs partially bridge this gap by segmenting BIM data by rooms, a fully integrated approach that aligns BIMSegGraphs with standard 3D scene graph structures remains a challenge. Lastly, while LLMs have shown significant potential in task planning and semantic interpretation, most research has focused on small-scale household applications. However, LLM-based robot task planning for domain-specific tasks in the AEC sector remains largely unexplored. Additionally, the performance and capability of LLMs in spatial and dimensional reasoning for large-scale AEC environments remain poorly understood.

Therefore, the research gaps can be summarized in the following research gaps:

2.4.1 Research Gaps

- **Lack of Research in Automation for Corrective Maintenance:**
  Existing studies predominantly target preventive maintenance due to its predictable nature, resulting in a shortage of research on frameworks and methods that can handle the dynamic and variable conditions of corrective maintenance tasks.

- **Need to Bridge 3D Scene Graphs with BIMSegGraphs:**
  Conventional 3D scene graphs in computer vision lack essential building elements, whereas BIM-derived graphs emphasize volumetric, structurally oriented data. Although BIMSegGraphs have addressed the mismatch between BIM and robotics by reconstructing BIM data on a room-by-room basis, there remains a need to bridge 3D scene graphs between BIMSegGraphs to enable full integration from BIM to robotics.

- **Limited Evaluation and Analysis of LLM Spatial Reasoning for the AEC Domain:**
  Research on LLM task planning has largely focused on the household domain, with relatively few studies addressing tasks in the AEC domain. In particular, the performance evaluation of LLMs in spatial reasoning and dimensional reasoning for the AEC domain is insufficient, highlighting the need for further analysis and evaluation.

Chapter 3

Methodology

3.1 Methodology Overview

This chapter begins by revisiting the research questions and objectives, providing an overview of the entire framework pipeline. More specifically, in section 3.2, I outline the key data sources and preparation methods used in this study, then introduce the detailed pipeline of the framework and its five main LLMs-based modules in section 3.3. Finally, I present the evaluation methodology, including the assessment framework and visualization tools in section 3.4.

This research aims to develop an initial framework for automating corrective maintenance tasks tailored to robots. Corrective maintenance requires immediate responses to unexpected defects and dynamic conditions, making it more challenging to automate compared to preventive maintenance. To address these challenges, this research leverages LLMs, which can interpret diverse scenarios and apply repair domain knowledge to a certain extent.

However, using LLMs and robots to automate corrective maintenance tasks also demands a standardized way of representing 3D environments that aligns both the AEC domain and robotics. In robotic applications, 3D Scene Graphs have emerged as a new method to communicate large-scale 3D building environments to LLMs, yet their current structures do not fully account for essential AEC domain requirements, such as incorporating fundamental building elements like walls, ceilings, and floors.

Meanwhile, existing techniques for converting BIM to graph-based formats in the AEC domain tend to focus on volumetric data rather than the surface data that robots directly interact with inside rooms. While BIMSegGraphs have partially addressed this gap by aligning scanning data with BIM, further work is needed to integrate these BIM-driven graphs into the conventional 3D Scene Graphs. Ultimately, this research aims to propose a fully end-to-end methodology, encompassing data preparation, 3D environment modeling, and LLMs integration with robotic libraries for high-level task planning, to facilitate automated corrective maintenance in real-world building environments.

Moreover, although LLM-based robot task planning has achieved notable milestones for household tasks, the spatial and dimensional reasoning capabilities required for AEC applications remain underexplored. Another challenge lies in developing an effective methodology to seamlessly integrate BIM data with existing robotic libraries. Therefore, the proposed methodology focuses on addressing the following key problems:

- Developing an end-to-end high-level robotic task planning framework that combines LLMs and BIM-driven 3D Scene Graphs
- Bridging the gap between BIMSegGraphs and conventional 3D Scene Graphs for robotic applications in building maintenance
- Integrating the LLM-based task planning framework with standard robotic libraries to streamline automated building maintenance

3.1.1 Framework Overview

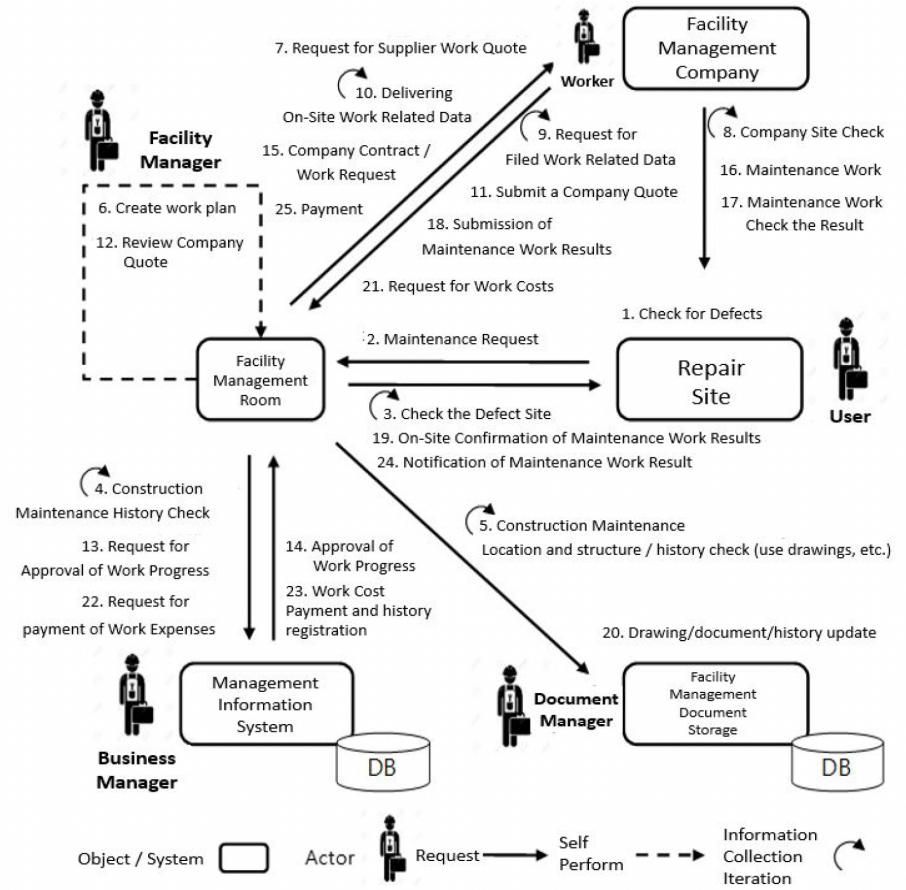


Figure 3.1: Corrective Maintenance Process from (Park et al., 2021)

As illustrated in fig. 3.1 (Park et al., 2021), the corrective maintenance process, ranging from defect reporting to on-site repairs, consists of multiple intermediate steps such as defect inspection, repair planning, data collection and analysis, and document review and updating. Each of these steps involves complex and interconnected tasks. To fully automate this corrective maintenance process, four key automated components are needed: (1) automated report interpretation, (2) automated mission planning (3) automated data collection and analysis, and (4) automated documentation.

In this study, I focus on automating the first two components through the use of LLMs. Due to time and resource constraints, this research is limited to text-based defect reports. However, this framework could be extended to incorporate visual resources such as

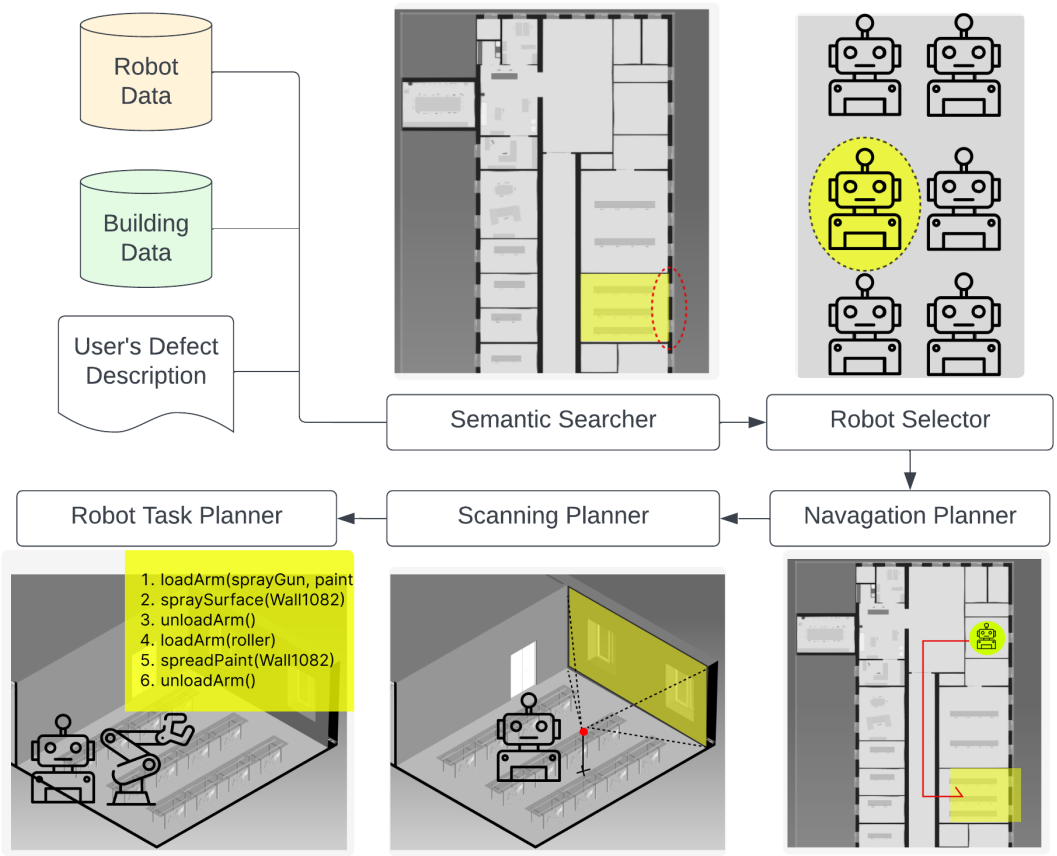


Figure 3.2: The Framework Overview

images by using Visual-Language Models to convert them into textual descriptions, thereby integrating them into this LLMs-based workflow.

As shown in fig. 3.2, the proposed framework is composed of five key LLMs-based modules designed to process various corrective defect cases described in natural language reports. The initial inputs to these modules include user-generated defect reports, building data in BIM-driven 3D Scene Graphs, and a robot database containing multiple robot configurations. Each modules extracts and utilizes the relevant information from these inputs to accomplish its designed tasks.

First, the Semantic Searcher module identifies the defect object node and the corresponding room node from the BIM-driven 3D Scene Graphs based on the user's report. Next, the Robot Selector module chooses the most suitable robot from the robot database to handle the specified defect task. After selecting an appropriate robot, the Navigation Planner determines a navigation path from the robot's current position to the defect's location. Upon arrival in the defect room, the Scanning Planner proposes the optimal camera position and orientation for defect scanning, taking into account spatial coordinates. Finally, the Robot Task Planner uses the selected robot configuration and spatial information to generate a sequence of actions required to perform the repair task.

In summary, this LLMs-based framework automates the entire planning stage of the corrective maintenance process: from interpreting the defect report, identifying the defect's location, selecting a suitable robot, determining the navigation path, planning the scanning strategy, and generating a comprehensive robot task plan.

3.2 Data Preparation

In this section, I introduce the three main types of data in the proposed framework: user input, building data, and robot data, and explain how each type is defined and processed. In section 3.2.1, I discuss what information should be provided from users as an initial defect report, which is transferred in text. In section 3.2.2, I explain the scale of the 3D environment used in this research and describe the process of converting the BIM model into BIM3DSG. This BIM3DSG is newly introduced in this research by bridging the gaps between the BIMSegGraphs (Claire, 2024) and 3D Scene Graphs (Armeni et al., 2019), as further detailed in section 3.2.2.1. Finally, section 3.2.3 explains how robot data is structured and configured for this framework.

3.2.1 User Input

The main difference between corrective and preventive maintenance processes lies in how repair requests are initiated. Corrective maintenance begins with a defect report from users, making the interpretation and understanding of such user descriptions of defects crucial for automation. Since defects in corrective maintenance occur unexpectedly, they are typically described by users in various ways. This characteristic highlights the potential of LLMs for their strengths in natural language processing. However, due to its countless variations, this framework establishes a minimum set of information required for further processing in subsequent modules, as shown in code 3.1:

- **Defect type:**
  Information that helps infer the nature of the defect (e.g., crack, cleaning, painting).
- **Affected building element:**
  Identification of the building element where the defect has occurred (e.g., wall, ceiling, floor).
- **Defect location:**
  Details about the room where the defect is located.

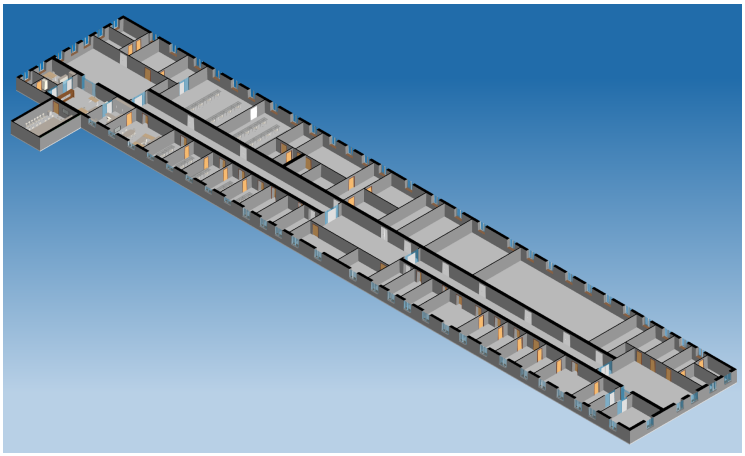
code 3.1: User Input Examples

```
#simple user input example:
"The largest door in room ID 9 is damaged."
#complex user input example
"The wall with one window in the room directly to the right of the
   entrance to room ID 26 has a crack"
```

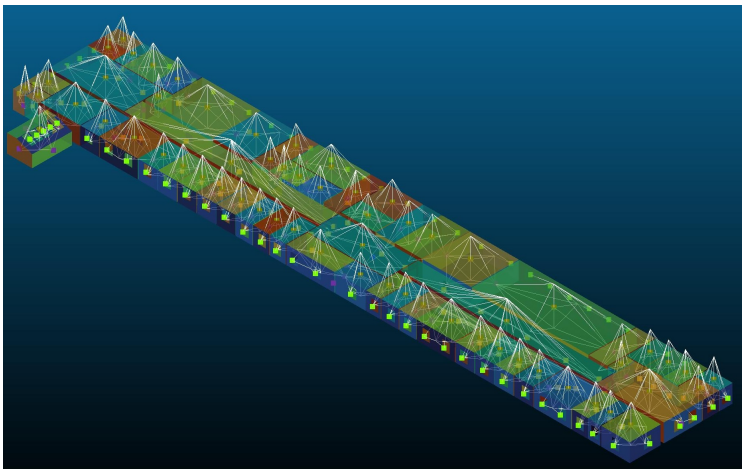
3.2.2 Building Data

The 3D Environment used in this study is based on BIM representation of a single floor of a Technical University of Munich (TUM) building. This BIM model, created using Autodesk Revit, contains 62 rooms and six different-sized doors to accommodate diverse use cases when evaluating the LLM-based modules. Before converting the BIM model into a 3D Scene Graph format, it was necessary to reorganize the BIM data by room. This step addresses the issue noted in section 2.4.1, where commonly used 3D Scene Graphs, structured hierarchically by rooms in robotics, differ from the object-oriented definition of BIM models.

To achieve this, the BIM-Room-Graph framework (Claire, 2024) was utilized. Before applying the module, the BIM model was first converted into the IFC format, as this serves as a prerequisite. The module then uses a geometry kernel and concepts from collision detection to reorganize the BIM model's surfaces into room-based groups. As a result, each BIM element is broken down into individual surfaces, which are subsequently assigned to their respective rooms, as illustrated in fig. 3.3b. Finally, the reorganized data is exported as GraphML and CSV files, with NetworkX used for graph representation.



(a) 3D Environment represented in BIM Model



(b) The Result of the BIM-Room-Graph framework

Figure 3.3: Comparison of 3D BIM representations and their room-based results

As illustrated in fig. 3.3b, the BIM-Room-Graph framework provides each surface and element (such as window and doors) with attributes including the center point coordinate, geometry information derived from Principal Component Analysis (PCA), normal vector, element type, associated room and IFC Global Unique IDentifier (GUID) to their respective rooms, as shown in code A.1. With the BIM model data now structured on a room-by-room basis, the next step involves remapping this data into the BIM3DSG outlined in the following section, ultimately producing the final 3D scene graphs.

3.2.2.1 BIM-driven 3D Scene Graphs (BIM3DSG)

While the 3D Scene Graph structure proposed by Armeni et al. (2019) provides a foundation for spatial relationships in computer vision and robotics, it does not incorporate essential building elements crucial for the AEC domain, such as walls, ceilings, floors, windows, and doors. In contrast, BIMSegGraphs (Claire, 2024) offers a method to reconstruct BIM data on a room-by-room basis by segmenting surfaces. However, this approach remains insufficient for fully integrating BIM data into robotic applications. To address this gap, I propose BIM3DSG, which extends BIMSegGraphs to 3D Scene Graphs Structure by adapting the format and unifying terminology. I first compare BIMSegGraphs (Claire, 2024) and Sayplan's 3D Scene Graphs (Rana et al., 2023) to highlight their differences, then present the newly proposed BIM3DSG structure that bridges both approaches for robotic applications using BIM.

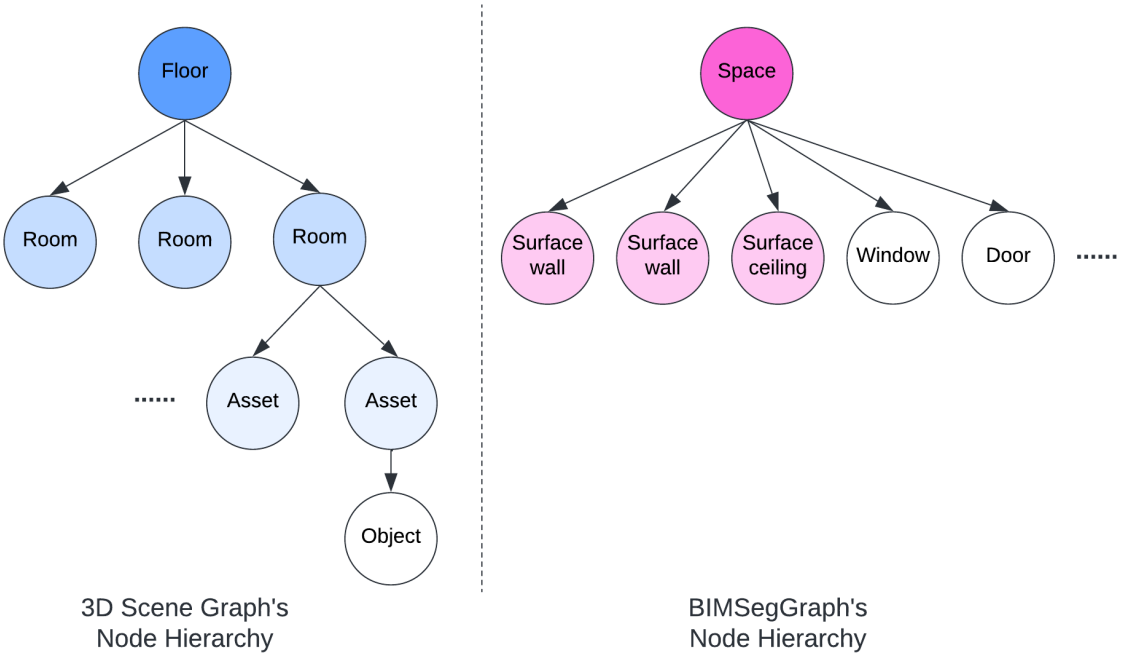


Figure 3.4: Node Comparison between BIMSegGraph and SayPlan's 3D Scene Graph

When comparing the node structure between BIMSegGraphs and Sayplan's 3D Scene Graphs illustrated in fig. 3.4, Sayplan's structure introduces a distinct node hierarchy. In this hierarchy, Asset nodes represent immovable objects such as closets, fridges, and drawers, while Object nodes refer to movable items that can interact with robots. Additionally, Pose

nodes, placed at the same level as Room nodes, are used to provide scanning data for the 3D Scene Graph, similar to the approach introduced in the initial 3D Scene Graphs by Armeni et al. (2019).

In contrast, BIMSegGraphs have hierarchical relationships only between rooms and other elements through containment relationships, where elements contained within a room have a relationship with that room. However, BIMSegGraphs still maintains the distinction between building structure surface nodes and building component nodes such as windows and doors through adjacency relationships.

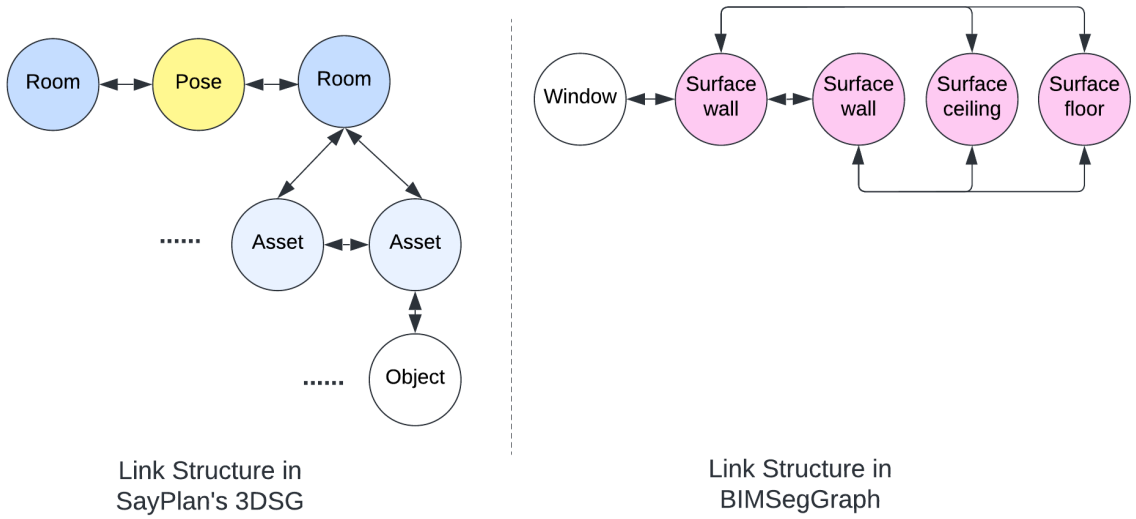


Figure 3.5: Link Comparison between BIMSegGraph and SayPlan's 3D Scene Graph

Regarding the link structure, shown in fig. 3.5, Sayplan's 3D Scene Graph maintains edges in a separate Links layer from the Node layer, representing connections between elements at the same level or adjacent hierarchical levels. Room connections are specifically described through Pose nodes (e.g., [kitchen<->pose1][pose1<->tom's room]), only serving semantic searching, while Dijkstra algorithm-based navigation planning.

However, BIMSegGraphs generates links based on three types of spatial relationships: adjacency, intersection, and containment between elements. The adjacency relationships primarily capture connections between structural elements and components, where surfaces have relationships with adjacent surfaces, and elements like windows and doors have relationships with their hosting surfaces. While intersection relationships are particularly important for Mechanical, Electrical, and Plumbing (MEP) elements passing through spaces, and containment relationships specifically represent space-element relationships. However, these relations do not have hierarchies because BIMSegGraphs is an efficiently compressed structure for matching the scanning data to BIM model.

To summarize, conventional 3D Scene Graphs typically employ a detailed hierarchical structure centered on objects that robots can interact with. This approach excels at specifying complex relationships among movable objects. In contrast, BIMSegGraphs adopts a more flattened structure that focuses on building surfaces, such as walls, ceilings, and floors, capturing adjacency relationships based on their room-level organization.

Although BIMSegGraphs effectively represents a building's key structural elements in the AEC domain, it lacks the detailed hierarchical relationships found in conventional 3D Scene Graphs.

To combine the strengths of both methods, I propose BIM3DSG, which maps the necessary AEC elements from BIMSegGraphs into the hierarchical structure of 3D Scene Graphs. Specifically, BIM3DSG extends the concept of rooms to encompass more general spaces, including hallways and other non-room areas. It renames the Object layer to the Surface layer and introduces a distinct Component layer for replaceable building elements, such as windows and doors, following the representation style of BIMSegGraphs for 3D environments.

While SayPlan's 3D Scene Graph stores hierarchical information in both node attributes and links, BIM3DSG avoids such duplication by recording hierarchical relationships only within the node layer's attributes, as illustrated in code A.3. This design separates purely navigational information (such as door-based connections between spaces) into the link layer. Consequently, the link layer in BIM3DSG supports efficient navigation planning, while the node layer manages the building's hierarchical structure. This separation significantly reduces token usage for LLM-based planning modules, as unnecessary adjacency and containment details do not add extra complexity to the graph.

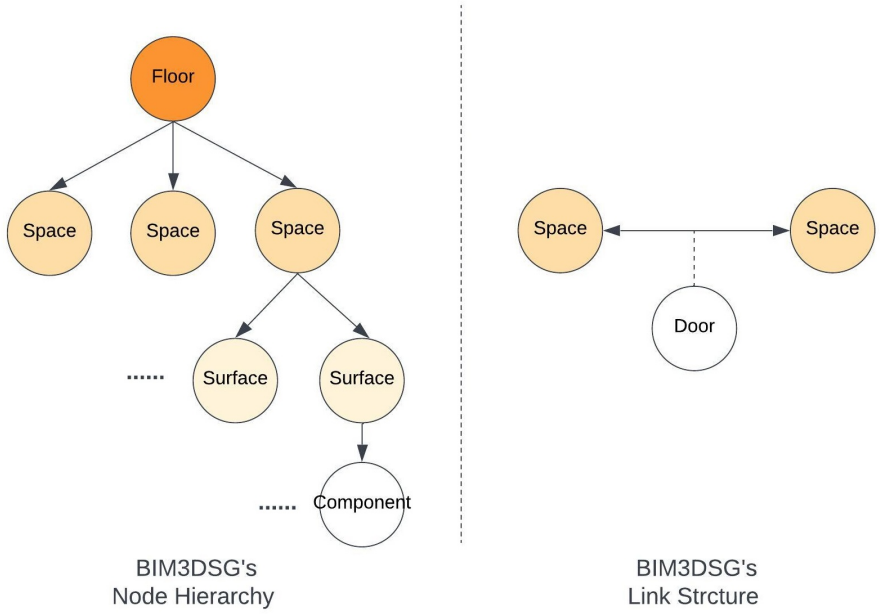


Figure 3.6: BIM3DSG Structure

Finally as shown in fig. 3.6, BIM3DSG forms a hierarchical structure of Floor → Space → Surface → Component, however, the Floor layer was omitted in this study since it deals with a single floor. Each layer shares common attributes: ID, location, and size. The location attribute specifies the central coordinate of each element, while the size attribute represents absolute positive values for width, length, and height. This notation follows the initial 3D scene graph structure of Armeni et al. (2019). For quick identification, I introduce the ID range as follows: spaces use IDs from 0 to 1000, walls start from 1000, ceilings

start from 2000, floors start from 3000, windows start from 4000, and doors start from 5000.

As shown in table 3.1, the final BIM3DSG consists of 579 nodes (62 space, 372 surface, and 145 component nodes) and 72 links. The complete graph representation comprises 35,987 tokens, with 34,834 tokens for nodes and 1,153 tokens for links.

Table 3.1: Node Numbers and Token Sizes of BIM3DSG

Entity Type	Number of Entities	Total Number of Tokens
Space Nodes	62	4,476
Surface Nodes	372	21,666
Component Nodes	145	8,685
Total Nodes	579	34,834
Total Links	72	1,153
Full Graph	651	35,987

3.2.3 Robot Data

The robot data definition in this framework builds upon the work of (Kannan et al., 2024), which proposed a framework utilizing LLMs for task planning across multiple robots. To adapt Kannan et al. (2024)'s framework for this research, I began by defining the scope of basic building corrective maintenance tasks: (1) cleaning surfaces, (2) painting surfaces, (3) disinfecting elements, (4) filling cracks and (5) picking up trash.

3.2.3.1 Robot definition

Among currently available robots, movable one-armed robots are considered the most feasible option for handling complex and detailed maintenance tasks. Each robot is characterized by attributes such as id, actions, equipment, materials, maximum reach height, and camera configuration, as shown in code 3.2.

code 3.2: Robot Configuration Example

```
{
    "id": 1,
    "actions": [
        "loadArm",
        "unloadArm",
        "spraySurface",
        "wipeSurface"
    ],
```

```
    "equipments": [
        "sprayGun",
        "wiper"
    ],
    "materials": [
        "cleaningSolution"
    ],
    "max_reach_height": 3000,
    "camera": {"FOV": 90}
},
```

3.2.3.2 Action

To perform diverse tasks, each one-armed robot must load and unload different equipment and materials onto its single arm. While each robot references only the action names, the detailed parameters and preconditions for each action are explicitly defined in the action specification. These preconditions were included to provide concise and explicit guidelines for LLMs-based planner to generate valid action sequences. The action specifications ensure clarity by using logical operators, enabling LLMs-base planner to interpret and sequence actions accurately while adhering to task requirements. The following actions were identified to cover the five basic corrective maintenance tasks, as shown in code A.2:

code 3.3: Action Specification Example

```
[
  {
    "action": "loadArm",
    "parameters": ["equipment", "material=None"],
    "preconditions": [
      "if␣equipment␣==␣'sprayGun'␣and␣material␣!=␣None"
    ]
  },
  {
    "action": "unloadArm",
    "parameters": []
  },
  {
    "action": "collectTrashes",
    "parameters": [],
    "preconditions": [
      "equipment␣==␣'gripper'",
      "action␣==␣'scanTrashes'"
    ]
  },
  ...
]
```

3.2.3.3 Equipment and Material

To execute various tasks and actions, the robot can integrate with diverse equipment such as the Spray Gun, Scrapper, and Gripper. The Spray Gun, in particular, can load materials like cleaning solutions, disinfectants, paint, and filler to perform basic corrective maintenance tasks, as shown in code 3.4.

code 3.4: Euqipment andMaterial Examples

```
"equipments" = [
    "sprayGun",
    "wiper",
    "scraper",
    "gripper",
]

"materials" = [
    "cleaningSolution",
    "disinfectant",
    "paint",
    "filler"
]
```

3.2.3.4 Final Robot Database

The final robot database is provided in JSON format. It includes robot specifications, the location of the robot's storage area, and each robot's individual configuration, as shown in code 3.5.

code 3.5: Robot Database in JSON

```
{
    "robot_specifications": {
        "robot_type": "one-arm␣robot",
        "actions": actions,
        "equipments": equipments,
        "materials": materials,
    },
    "robot_storage": {
        "room_id": 46
    },
    "robots": robots,
}
```

3.3 LLM-based Modules

This section explores why certain LLMs models were chosen for this research and explains the architecture of the framework. First, I briefly discuss the selection criteria and characteristics of the chosen LLM Models. Then, I examine the framework's pipeline, which consists of five LLM-based modules with three primary databases.

3.3.1 LLM Model Choice

In selecting the LLM for the framework, I primarily used GPT-4o as the base model, as it was OpenAI's most advanced model as of October 2024. This choice was motivated by its well-documented performance and widespread use in LLMs research, making it an ideal model for evaluating spatial their inherent reasoning capabilities of LLMs. Additionally, I partially tested the o1-preview model, available from November 20. 2024, for cases where GPT-4o showed limitations and for more complex tasks. This decision was influenced by cost considerations. The o1-preview model represents a new text-based model of LLMs trained through reinforcement learning, specifically designed for complex reasoning tasks. It employs a unique approach where the model develops an internal chain of thought before providing responses, using new reasoning tokens (OpenAI, 2024). While this can result in higher operational costs regardless of input size, initial tests with the o1-preview chatbot demonstrated remarkably high accuracy in complex spatial reasoning tasks, leading me to include it in the final API-based evaluation as well.

3.3.2 Framework Architecture

The pipeline of the framework, shown in fig. 3.7, consists of three primary databases and five LLM-based modules. The framework utilizes three main data resources: (1) User Reports, (2) Building Data in BIM3DSG, and (3) Robot database. Since these databases can easily exceed GPT-4's token limitations for inputs and outputs, the LLM-based modules process only specific subsets of the database, while their outputs are restricted to generate IDs or brief responses. These subsets are retrieved by local functions using generated IDs or through direct access to the database. Based on this approach, the framework consists of five main LLM-based modules for robotic planning:

1. **Semantic Searcher**: the searcher to identify relevant defect node IDs and room IDs within the BIM3DSG database by analyzing natural language user reports
2. **Robot Selector**: the robot selector to determine the most appropriate robot from the database based on the user's report and the defect's detailed information.
3. **Navigation Planner**: the path planner to generate the path with a sequential path through rooms and doors, from the robot's storage location to the room containing the defect.

4. **Scanning Planner**: the planner to calculate the optimal camera position and orientation to scan the defect object within the defect's room.

5. **Robot Task Planner**: the planner to generate the robot's action sequence to perform the maintenance tasks based on the selected robot's configuration.

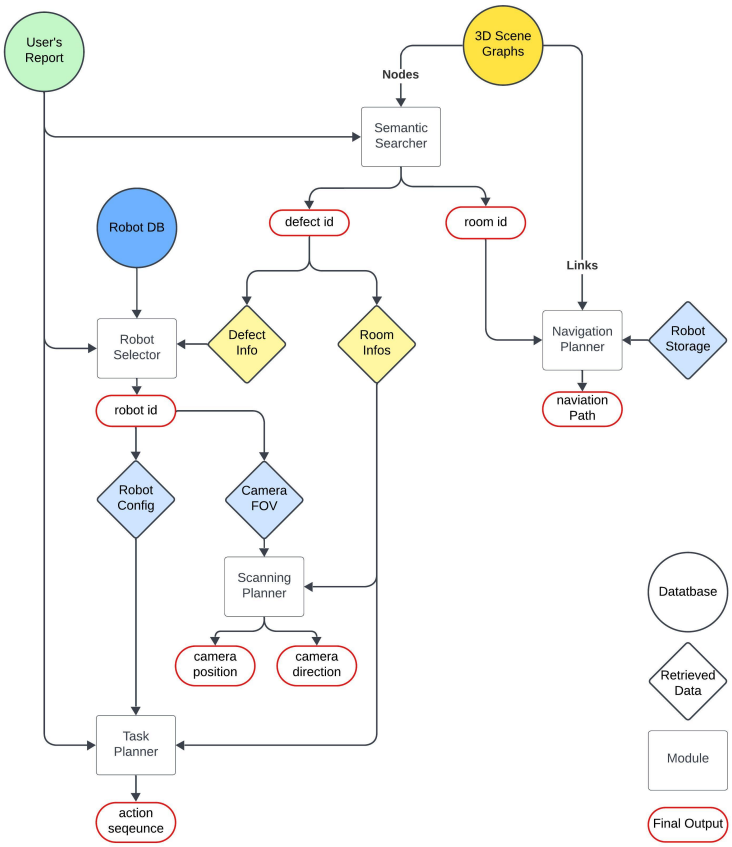


Figure 3.7: Data Flow of the Framework

The proposed framework operates as follows. First, the Semantic Searcher module processes the BIM3DSG data along with the user's report to identify the defect node ID and the room node ID where the defect is located. Using the defect node ID, the framework retrieves two types of node information: (1) detailed attributes of the defect node, and (2) detailed attributes of all building elements nodes within the defect's room. This room environment information is provided to the Scanning Planner and Robot Task Planner, which generate plans within the given environmental condition.

Next, the Robot Selector analyzes the user report with the retrieved defect's details such as size and location to choose the most suitable robot. Once the optimal robot is identified, the framework retrieves the selected robot's configuration and camera Field of View (FOV) from the robot database, which are then provided to the Robot Task Planner and Scanning Planner.

Following robot selection, the Navigation Planner calculates the shortest path from the selected robot's storage room to the defect's room. This path is represented as a sequence of room and door IDs, effectively guiding the robot to the target environment.

With the environment and path established, the Scanning Planner determines the optimal camera position and orientations for scanning the defect. To do so, it takes into account the coordinates, dimensions, and spatial arrangement of all building elements within the defect's room.

Finally, The Robot Task Planner generates a sequence of actions for executing the required maintenance task with the user's defect report, the retrieved room environment details, and the selected robot's configuration including robot action capabilities.

While the Robot Selector and Navigation Planner, as well as the Scanning Planner and Robot Task Planner, are technically independent in their sequence, they are executed sequentially to extend the framework in the future and align with the maintenance process workflow.

3.4 Evaluation Methods

This section outlines the methodologies used to evaluate the proposed LLM-based modules, including both semantic and computational tasks. It provides details on the experimental setup, the classification of evaluation criteria according to task complexity, and the specific test conditions designed for each module. Moreover, it introduces the visualization tools developed to assess spatial and dimensional reasoning tasks, highlighting the importance of qualitative and quantitative analysis for understanding the performance and limitations of the evaluated modules.

3.4.1 Experimental Setup

In the evaluation framework, I utilized OpenAI API models, GPT-4o and o1-preview. The GPT-4o models served as the primary baseline for assessing the performance of each module, while the o1-preview model was employed selectively, only when the GPT-4o model exhibited failure. This selective usage was motivated by cost considerations. For the experimental configuration, while setting the temperature parameter to 0 could enhance response consistency, the o1-preview reasoning model (as of November 2024) does not yet provide temperature API control (OpenAI, 2024). Therefore, to ensure a fair comparison between models, the GPT-4o model's temperature was set to its default value of 1 (OpenAI, 2024). For each use case, up to five experimental trials were conducted, with outcomes categorized as success, failure, or intermediate success. The evaluation included both individual assessments of each module as well as integrated, end-to-end tests conducted across representative scenarios involving all modules.

Each module is characterized by its inputs, outputs, and task of complexity, as summarized in table 3.2. The measure of task complexity is introduced to provide context for interpreting module performance, as accuracy rates can vary based on task difficulty. This complexity measure indicates the relative difficulty of each LLM's reasoning task by comparing it to the time and effort a human would require to perform the same task.

Table 3.2: LLM-based Modules Configurations

Module	**Task Complexity**	**Inputs**	**Outputs**
Semantic Searcher	High	User Report, BIM3DSG[Nodes]	Defect Node ID, Room Node ID
Robot Selector	Low	User Report, Defect Details, Robot Database	Robot ID
Navigation Planner	High	Defect Room ID, Robot Storage Room ID, BIM3DSG[Links]	A Sequence of Room and Door IDs
Scanning Planner	Very High	Room Env Details, Robot's Camera FOV	Camera Position Camera Orientation
Task Planner	Mid	User Report, Room Env Details, Selected Robot's Config	A Sequence of Robot Actions

3.4.1.1 Task Complexity and Evaluation Purpose

1. **Semantic Searcher** (Evaluation of LLM's Spatial Graph Querying)
   The semantic searcher module assesses the LLM's ability to perform spatial graph queries. Given a complex 3D environment represented by BIM3DSG (see table 3.1), the model is required to interpret and query data based on a user's natural language report. Specifically, it must identify defect nodes and room nodes that match the user's description. The input graph contains 579 nodes in total, making the correct identification of relevant nodes hard. Since a human would find searching through such a large dataset time-consuming and challenging, I label the complexity of this task as high.

2. **Robot Selector** (Evaluation of LLM's Simple Multi-Reasoning)
   The robot selector module tests the LLM's capacity for simple multi-reasoning, including basic semantic understanding, and database queries related to a limited set of maintenance robots. Given a set of 10 robots (e.g. a cleaning bot, a disinfecting bot, or a multi-task bot), the model must determine which robot is most suitable for a given maintenance task. Additionally, it must consider a simple dimensional comparison between a defect's location and a robot's maximum reachable height.

Since the search space is limited and relatively straightforward, the complexity of this task is low.

3. **Navigation Planner** (Evaluation of LLM's Path Planning)
   The navigation planner module evaluates the LLM's ability to perform path-planning tasks. This involves distance calculations and understanding spatial connections within the scene graph. As shown in table 3.1, there are 62 room nodes, and the model must identify the shortest path among them. To navigate such complex building environments, graph searching algorithms like A* or Dijkstra's are essential for efficient path planning. Therefore, I determined this task to be high complexity because of these challenges.

4. **Scanning Planner** (Evaluation of LLM's Dimensional Reasoning)
   The scanning planner module evaluates the LLM's dimensional reasoning capabilities. Using the provided building element coordinates and size data of a room, the task involves calculating the optimal scanning position and orientation. This requires reconstructing the 3D environment based on the given data, ensuring a certain distance is maintained from the defect location, and making sure the defect area is within the camera's FOV while also considering angles and surface normals. Due to these complexities, which can be challenging even for humans, the complexity level is classified as Very High.

5. **Task Planner** (Evaluation of LLM's Building Maintenance Domain Knowledge)
   The task planner module tests the LLM's building maintenance domain knowledge. Given a user report describing a particular defect scenario, the model must generate a plausible sequence of maintenance actions that reflect a sensible, human-like understanding of the required steps. Although not as extensive as the semantic searcher or navigation planner in terms of data size and complexity, this module still demands domain-specific reasoning, which I classify as mid in complexity.

3.4.2 Evaluation Classification and Criteria

To evaluate the modules based on their different inputs, I classify them into two types: (1) Semantic Modules and (2) Computational Modules as shown in fig. 3.8. Semantic Modules include modules that require semantic interpretation of a user's report in natural language to perform their task, such as Semantic Searcher, Robot Selector, and Task Planner. To evaluate these Semantic Modules, I tested them with different semantic types of user reports in their module testing. For Computational Modules, including Navigation Planner and Scanning Planner, which only require computational calculation and do not process semantic information such as user reports, I performed quantitative evaluations and visualized their results in 2D and 3D virtual environments.

Both Semantic Modules and Computational Modules were tested with various use cases. For the Semantic Modules, the Semantic Searcher module was evaluated by categorizing user queries into simple searches without spatial reasoning and more complex searches

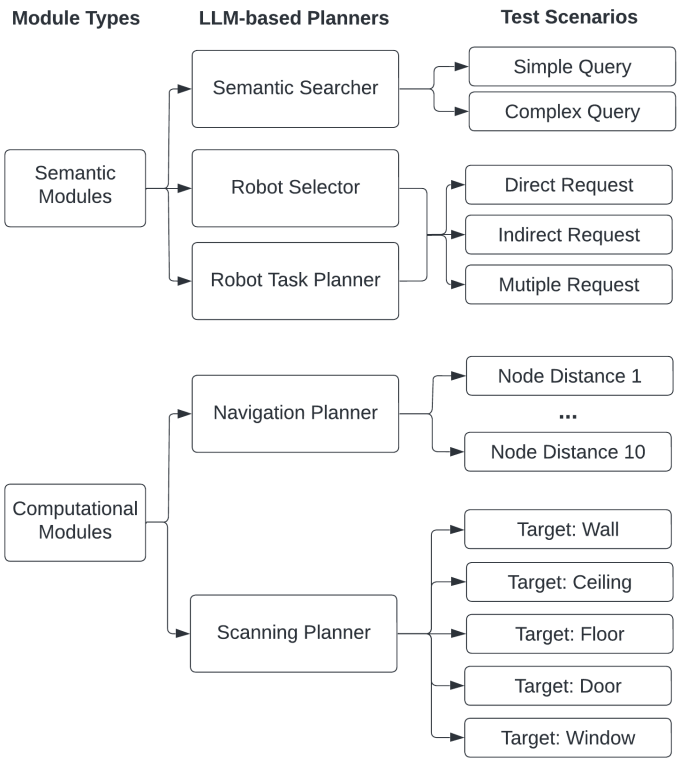


Figure 3.8: Module Evaluation Classification

involving spatial reasoning, following the evaluation criteria proposed by SayPlan, a pioneer researcher of LLM-based semantic searchers with 3D scene graphs (Rana et al., 2023). Since both the Robot Selector and Robot Task Planner modules must interpret the user's repair requests, they were tested with direct requests, indirect requests, and multiple requests.

For the Computational Modules, quantitative evaluations were conducted to assess their performance across various conditions. In the case of the Navigation Planner, tests were conducted according to node distance. Here, node distance refers to the shortest path represented as a room-door-room sequence, where a path passing through one door is considered to have a node distance of 1, and a path passing through two doors is considered to have a node distance of 2. This distinction was made based on initial test results indicating performance variations of the LLM depending on node distance. For the Scanning Planner, it was difficult to implement an automated algorithm for quantitative evaluation, so the results were visualized in three dimensions for each scanning target object (e.g., wall, ceiling, floor) and manually inspected by humans.

As mentioned in section 3.4.1, due to the lack of temperature API control in o1-preview and the corresponding use of default temperature value 1, All modules were tested to up to five attempts per test case. If a given task was successfully completed on the first attempt, the result was marked with an "O". If it succeeded within five attempts, a "△" was recorded along with the trial number on which it succeeded, and if it did not succeed within five attempts, it was marked with an "X". Definitions of success, intermediate success, and

failure were established individually for each module, reflecting differences in their testing objectives and configurations.

3.4.2.1 Ground Truth Criteria

Different ground truth criteria were established to evaluate each module's performance. To assess various use cases systematically, I implemented a hardcoded evaluation framework that required appropriate ground truth data for generating quantitative results. For the semantic searcher and robot selector modules that handle ID selection tasks, I manually annotated the correct IDs for each use case. Similarly, for robot action sequences, I manually created optimal action sequences as ground truth for each use case and evaluated the system's output by comparing them, and carefully annotating the results of executable plans. In assessing the navigation planner, I compared its generated paths against the optimal path computed by the A* algorithm and conducted a qualitative assessment through visualization. The scanning planning module presented a unique challenge, as pre-defining exact positions and orientations for all scenarios was impractical. To address this limitation, I employed a visualization-based qualitative assessment approach to evaluate the effectiveness of scanning strategies and their outcomes.

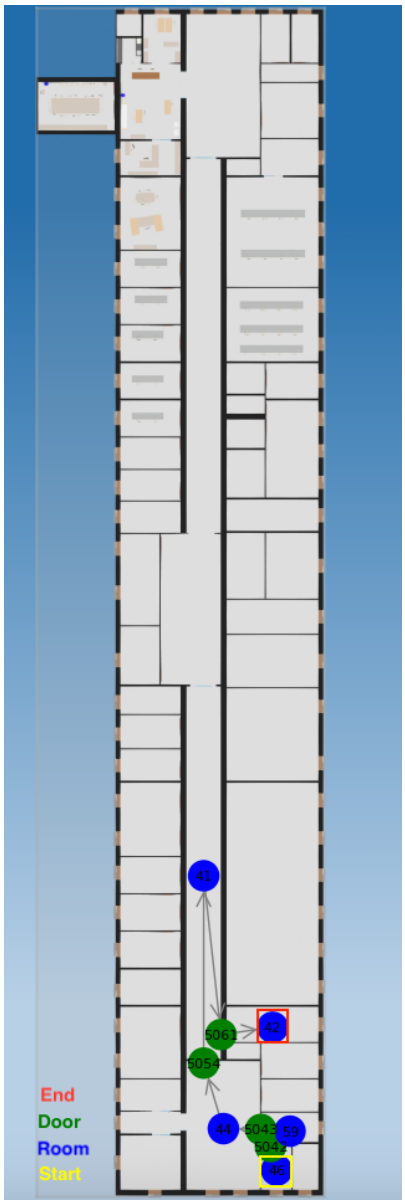
3.5 Visualization

This section introduces the visualization tools developed for the Navigation Planner and Scanning Planner. These modules are important for demonstrating the spatial and dimensional reasoning capabilities of LLMs. However, assessing the quality of these plans is challenging, as simple numerical metrics often fail to fully capture their effectiveness. For instance, in navigation planning, a proposed path may not be optimal but still executable, and visualization helps analyze details such as where hallucinations occur in complex spatial configurations. Similarly, evaluating the scanning planner involves understanding how well the scanning positions and orientations cover the defect area and whether they are optimal. Qualitative evaluation of these aspects is difficult, making visualization essential.

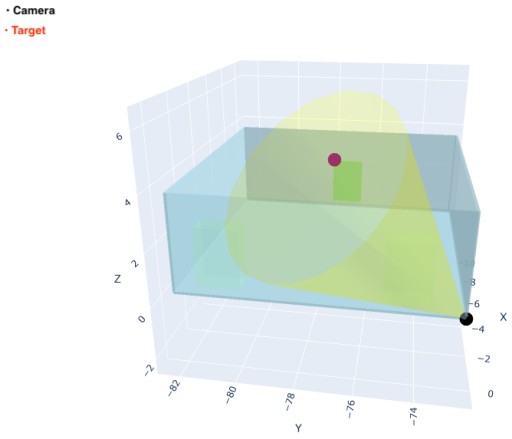
These visualization tools were designed to display planned paths and scanning positions/orientations, enabling qualitative and quantitative assessment. The developed tools provide the following features:

- **Navigation Path Visualizer:** Displays planned navigation paths and scanning positions, enabling both qualitative and quantitative assessments. Navigation paths are represented as graphs using NetworkX and overlaid on a 2D top-view plan of the 3D environment with Matplotlib, highlighting the starting point, destination, intermediate doors, and target objects, as shown in fig. 3.9a.

- **Scanning Planner Visualizer:** Visualizes the scanning position, orientation, and coverage area within a 3D environment using the Plotly library, while highlighting the defect object in the room, as shown in fig. 3.9b.



(a) Example visualization of navigation plan



(b) Example visualization of scanning plan

Chapter 4

Results

This chapter evaluates the proposed framework's performance across individual modules and the end-to-end pipeline including all modules. Tests were conducted on the semantic searcher, robot selector, task planner, navigation planner, and scanning planner using two different LLMs: GPT-4o and o1-preview. Section 4.1 presents the evaluation of individual modules, detailing the performance of semantic and computational modules, along with the detailed criteria for success, failure, and intermediate success for each module. Section 4.2 describes end-to-end tests conducted across five representative use cases, demonstrating how the modules perform together under varying conditions. Section 4.3 provides a qualitative analysis of the computational modules, utilizing 2D and 3D visualizations to assess results that cannot be fully captured through quantitative metrics. Finally, Section 4.4 summarizes the key findings and overall performance trends observed throughout the evaluation.

4.1 Module Results

This section shows the evaluation results for each module. The assessment includes three categories: (1) Success (O), (2) Failure (X), and (3) Intermediate Success (△). Success is marked when the module completes the task at the first trial. Failure occurs when the module cannot complete the task within the maximum, number of trials (five attempts). Intermediate success is noted when the module succeeds partially or fully within those five attempts with annotation of the trial number at which the module succeeded. The symbols used are:

- **O**: Success on the first attempt.
- △: Intermediate success within five attempts, with the successful trial number indicated.
- **X**: Failure to complete the task within five attempts.

4.1.1 Semantic Modules

As described in section 3.4.2, the semantic modules include the semantic searcher, robot selector, and task planner, all of which interpret user defect descriptions using semantic reasoning. The semantic searcher is evaluated with both simple and complex queries, while the robot selector and task planner are tested using direct, indirect, and multiple requests.

4.1.1.1 Semantic Searcher

The semantic searcher was tested using 20 user descriptions, categorized into simple and complex queries, as shown in table 4.1 and table 4.2. The symbol "O" is used when the module immediately finds the correct node in one attempt. If the module fails on the first attempt but succeeds within five attempts, the result is marked with "△", along with the trial number of success. If the module fails to identify the correct node within five attempts, the result is marked "X".

With GPT-4o as the base model, simple node searches, where no complex spatial reasoning is required, often succeeded on the first try, as shown in table 4.1. However, when the user request involved detailed attributes (e.g., type, location, size), the model sometimes did not succeed on the first attempt and showed mild hallucinations. For multiple-node queries, success often came around the fourth attempt, indicating more difficulty than with single-attribute queries. For complex node searches requiring spatial reasoning, performance decreased, as shown in table 4.2. The GPT-4o model succeeded on certain tasks, such as finding the largest door, but struggled with proximity reasoning, connections between rooms, relative positions (left/right), and inferring defect nodes indirectly. The model also failed when multiple types of reasoning were combined.

By contrast, the o1-preview model succeeded in these previously challenging cases on the first attempt, demonstrating stronger reasoning abilities. However, it required more computation time and cost per attempt.

4.1.1.2 Robot Selector

The robot selector faced relatively low task complexity. I tested the module with 15 different user descriptions categorized into direct robot requests, indirect specifications, multiple task requests, and height-based robot selections. In all 15 tests, it successfully identified the correct robot. When multiple candidate robots with similar capabilities were presented, it showed light hallucination even when the prompt instructed the model to choose the lower-cost option. Initially, the GPT-4o model selected the more capable robot first, choosing the lower-cost robot on the second attempt. In contrast, the o1-preview model selected the lower-cost robot on the first attempt, as shown in table 4.3.

4.1.1.3 Robot Task Planner

The robot task planner module, which has mid-level complexity, performed well on direct, indirect, and multiple-task requests by proposing a proper action sequence. In a trash-collection mission, for example, the GPT-4o model occasionally omitted an action (e.g., not loading a gripper) on the first attempt, but generated the correct answer on the second attempt. For multiple-task planning, although the suggested sequence was sometimes not perfectly ordered, it remained executable and was therefore considered an intermediate

success. The o1-preview model generated the correct answer immediately, whereas the base model showed hallucinations on its first attempt, as shown in table 4.4.

Table 4.1: Evaluation Results of the Semantic Searcher with Simple Queries

Semantic Type	**User Description**	**Trial**	**GPT-4o**	**Trial**	**o1-preview**
Search by ID	The wall with ID 1116 has a crack.	1	o[1]		
Search by child node ID	The wall with window ID 4055 needs to be painted.	1	o		
Search by type	The ceiling in room ID 32 is damaged.	2	△[2]	1	o
Search by location	The object located at [-26.1239, -7.47, 1.85] has a crack.	2	△	1	o
Search by size	The ceiling with the size [9.799, 23.86, 0.0] has a crack.	2	△	1	o
Search by parent-child	The wall that has a window has a crack in room 11.	1	o		
Search by parent-child	The wall with 8 doors in the room ID 6 needs to be painted.	1	o		
Negation condition	The wall that doesn't have any windows or doors in the room ID 12 has a crack.	1	o		
Combined conditions	The wall with two windows in the room located at [-5.15925, -33.5286, 7.85] needs repainting.	1	o		
Multiple nodes	The doors in room ID 8 need to be cleaned.	4	△	1	o

[1] Success at the first attempt

[2] Intermediate Success within five attempts

Table 4.2: Evaluation Results of the Semantic Searcher with Complex Queries

Semantic Type	User Description	Trial	GPT-4o	Trial	o1-preview
Dimension reasoning	The largest door in the room ID 9 is damaged.	1	o[1]		
Proximity reasoning	The wall in room ID 18 closest to the boundary with room ID 19 has a crack.	3	△[2]	1	o
Connection reasoning	The door in room ID 42 that connects to room ID 41 needs cleaning.	3	△	1	o
Orientation reasoning	The wall located on the north side in room ID 40 has mold.	4	△	1	o
Relative position	The wall to the immediate left after entering through door ID 5059 in room ID 39 has graffiti.	5	x[3]	1	o
Semantic reasoning	There seems to be water dripping from above in room ID 7, which may indicate a crack.	5	x	1	o
Semantic reasoning	It feels like cold air is coming from outside in the room ID 42.	5	x	1	o
Dimension + Orientation	The most northern door in the largest room is jammed.	5	x	1	o
Semantic + Dimension	Among the rooms with at least one door and one window, the window in the smallest room needs to be cleaned.	5	x	1	o
Orientation + Semantic	The wall with one window in the room directly to the right of the entrance to room ID 26 has a crack.	5	x	1	o

[1] Success at the first attempt

[2] Intermediate Success within five attempts

[3] Failure after five attempts

Table 4.3: Evaluation Results of the Robot Selector

Type	Semantic Type	User Description	Trial	GPT-4o	Trial	o1-preview
Direct	Cleaning	The floor needs to be wiped with cleaning solution.	1	o[1]		
	Disinfecting	The door handle needs to be disinfected.	1	o		
	Painting (low cost)	The wall needs to be painted.	2	△[2]	1	o
	Repair	The crack on the wall needs to be filled.	1	o		
	Trash Collection	The trashes on the floor need to be collected.	1	o		
Indirect	Cleaning	I dropped coffee on the floor.	1	o		
	Disinfecting	The door handle is contaminated with germs.	1	o		
	Painting (low cost)	There is a graffiti on the wall.	2	△	1	o
	Repair	There is a crack on the wall.	1	o		
	Trash Collection	There are trashes on the floor.	1	o		
Height-based	Repair	The crack on the ceiling needs to be filled.	1	o		
	Painting	The ceiling needs to be painted.	1	o		
Multiple Tasks	Repair & Painting	The wall needs to be filled and painted.	1	o		
	Trash Collection & Painting	The trash on the floor needs to be collected and then the wall needs to be painted.	1	o		
	Cleaning & Disinfecting	All windows in the room need to be cleaned and disinfected.	1	o		

[1] Success at the first attempt

[2] Intermediate Success within five attempts

Table 4.4: Evaluation Results of the Robot Task Planner

Type	Semantic Type	Description	Trial	GPT-4o	Trial	o1-preview
Direct	Cleaning	The floor needs to be wiped with cleaning solution.	1	o[1]		
	Disinfecting	The door handle needs to be disinfected.	1	o		
	Painting (low cost)	The wall needs to be painted.	1	o		
	Repair	The crack on the wall needs to be filled.	1	o		
	Trash Collection	The trashes on the floor need to be collected.	1	o		
Indirect	Cleaning	I dropped coffee on the floor.	1	o		
	Disinfecting	The door handle is contaminated with germs.	1	o		
	Painting (low cost)	There is a graffiti on the wall.	1	o		
	Repair	There is a crack on the wall.	1	o		
	Trash Collection	There are trashes on the floor.	2	△[2]	1	o
Height-based	Repair	The crack on the ceiling needs to be filled.	1	o		
	Painting	The ceiling needs to be painted.	1	o		
Multiple Tasks	Repair & Painting	The wall needs to be filled and painted.	1	o		
	Trash Collection & Painting	The trash on the floor needs to be collected and then the wall needs to be painted.	2	△	1	o
	Cleaning & Disinfecting	All windows in the room need to be cleaned and disinfected.	1	o		

[1] Success at the first attempt

[2] Intermediate Success within five attempts

4.1.2 Computational Modules

As described in section 3.4.2, the computational modules comprise the navigation planner and the scanning planner. These modules primarily rely on numerical data rather than semantic information. The navigation planner is evaluated with different node distances (i.e., the number of doors the robot must pass), ranging from one to ten. The scanning planner is tested on various target objects such as walls, ceilings, floors, doors, and windows. To support detailed qualitative analysis, their results are visualized in 2D and 3D.

4.1.2.1 Navigation Planner

The navigation planner represents a high-complexity module. Its complexity increases with the required path length, which is defined by node distance, which is the number of doors that must be passed through. For paths passing through one to three doors, the GPT-4o model generated optimal paths on the first attempt. For four- to five-door paths, hallucinations appeared, but the model still succeeded within five attempts. For paths requiring passing through six to ten doors, the model failed within five attempts. The o1-preview model produced correct paths on the first attempt even for paths passing through six to ten doors, demonstrating superior spatial reasoning but at a higher computational cost and time, as shown in table 4.5.

Table 4.5: Evaluation Results of the Navigation Planner

Node Distance	Trial	GPT-4o	Trial	o1-preview
1	1	o[1]	1	o
2	1	o	1	o
3	1	o	1	o
4	2	△[2]	1	o
5	3	△	1	o
6	5	x[3]	1	o
7	5	x	1	o
8	5	x	1	o
9	5	x	1	o
10	5	x	1	o

[1] Success at the first attempt

[2] Intermediate Success within five attempts

[3] Failure after five attempts

4.1.2.2 Scanning Planner

The scanning planner was tested with objects in the same room, such as floors, ceilings, walls, windows, and doors, with FOV of 90 degrees. The GPT-4o model rarely produced

an optimal scanning plan, but it suggested partially feasible plans. For planar surfaces (floor, ceiling, wall), orientation predictions were accurate, but the suggested positions for those objects often partially covered the defect's area. For objects like windows and doors, the base model, GPT-4o, did not provide optimal positions or orientations, yet still suggested partially effective solutions. On the other hand, The o1-preview model gave optimal scanning plans for ceilings, walls, and windows regarding position, orientation, and coverage. However, for floors, it failed on the first two attempts, and only on the third attempt did it propose a limited but workable solution. For doors located between two spaces, the model suggested scanning positions from another room, even though the prompt instructed it to propose a plan within the same room, as shown in table 4.6.

Table 4.6: Evaluation Results of the Scanning Planner

Model	Target Object	Trial	Position	Orientation	Coverage
GPT-4o	Floor	1	△[2]	o[1]	△
	Ceiling	1	△	o	△
	Wall	1	△	o	△
	Window	1	△	△	△
	Door	1	△	△	△
o1-preview	Floor	3	△	△	△
	Ceiling	1	o	o	o
	Wall	1	o	o	o
	Window	1	o	o	o
	Door	2	△	△	△

[1] Successfully Predict the optimal scanning parameter

[2] Suggests a partially feasible scanning parameter

4.2 End-to-End Results

For the end-to-end evaluation, five representative use cases were tested to assess the full pipeline. Both GPT-4o and o1-preview models were tested once for each scenario. If the initial attempt did not yield a feasible solution, the result was marked as intermediate (△). For simpler, direct requests, the GPT-4o model succeeded in all modules except the scanning planner. As queries became more complex, or as navigation paths grew longer, the GPT-4o model showed higher hallucination rates. When the semantic module failed to identify the correct node, the subsequent navigation and scanning planners also failed due to incorrect room information. The o1-preview model handled these complex node searches and navigation planning better than the GPT-4o model. However, it did not always provide optimal scanning positions. In multiple-object scanning tasks, it occasionally proposed a scanning plan for only a single element rather than all required elements, as shown in table 4.7.

Table 4.7: Evaluation Results of the End-to-End Pipeline

Model	User Description	Node Searcher	Robot Selection	Navigation Path	Scanning Plan	Robot Task
Semantic Type	The wall ID 1055 needs to be cleaned with a cleaning solution.	simple query with id	direct request for a single task	distance 2	wall	direct request for cleaning
GPT-4o		o	o	o	x/x	o
o1-preview		o	o	o	△[1]/o	o
Semantic Type	The paint of the wall with a window in the room ID 43 is worn out.	simple query with a child node's condition	indirect request for a single task	distance 3	wall	indirect request for painting
GPT-4o		o	o	o	o/o	△
o1-preview		o	o	o	△[1]/o	o
Semantic Type	The floor in the largest room has a crack	complex query with dimension reasoning	indirect request for a single task	distance 4	floor	indirect request for filing
GPT-4o		x	o	x	△/o	o
o1-preview		o	o	o	x/x[2]	o
Semantic Type	The door handle in room ID 13 that connects to room ID 14 is contaminated with germs.	complex query with connection reasoning	indirect request for a single task	distance 6	door	indirect request for painting
GPT-4o		o	o	o	x[3]/△	o
o1-preview		o	o	o	△/△	o

Continued on next page

Table 4.7: Evaluation Results of the End-to-End Pipeline (Continued)

Semantic Type	all walls in room ID 32 need to be filled and painted again.	multiple query with a parent node's condition	direct request for multiple tasks	distance 9	4 walls	direct request for painting and disinfection
GPT-4o		o	o	x	△/△[4]	△
o1-preview		o	o	o	△/△[4]	△

[1] Predict the position not covering edge area

[2] Predict the not feasible scanning parameter

[3] Predict the position out of the room

[4] Predict a position for a single element

4.3 Visualization Analysis

As explained in section 3.5, quantitative metrics alone are insufficient for evaluating the navigation and scanning planners. Their outputs often involve spatial reasoning and coverage patterns that cannot be fully captured by numerical data. To address this, custom visualization tools were developed, allowing for a more qualitative examination of the computational modules' performance.

4.3.1 Navigation Plans

As shown in fig. 4.1, the GPT-4o model successfully generated optimal paths for node distances of up to 3 on the first attempt. For these shorter paths, the model consistently predicted the correct sequence of rooms and doors, demonstrating strong performance in simple navigation tasks.

However, starting from node distances of 4 to 5, the model began to exhibit hallucinations. While it often predicted the initial segments correctly, it occasionally deviated from the optimal route, introducing errors in the later parts of the path. This trend became more pronounced as the node distance increased. For node distances of 6 and beyond, GPT-4o failed to generate successful paths within the maximum of five attempts. Although the model accurately identified the starting and ending points, it frequently skipped intermediate nodes, jumping directly to the final destination. This resulted in incomplete paths that were not viable for practical navigation tasks.

In contrast, the o1-preview model consistently produced correct paths across all tested node distances. It maintained accuracy even for distances of six to ten, suggesting stronger spatial reasoning and multi-step planning capabilities.

Overall, while GPT-4o performs well for shorter paths, its reliability decreases with more complex navigation tasks, making o1-preview the more suitable choice for longer routes and intricate environments.

4.3.2 Scanning Plans

As shown in fig. 4.3 and fig. 4.4, there are differences in the dimensional reasoning and scanning coverage between the GPT-4o and o1-preview models. When scanning basic building components such as walls, ceilings, and floors, GPT-4o generally predicts the correct orientation for scanning but fails to achieve full coverage of the target surface. This often results in partial scans that leave certain areas unaddressed, leading to assessments of intermediate success (△) in performance evaluations.

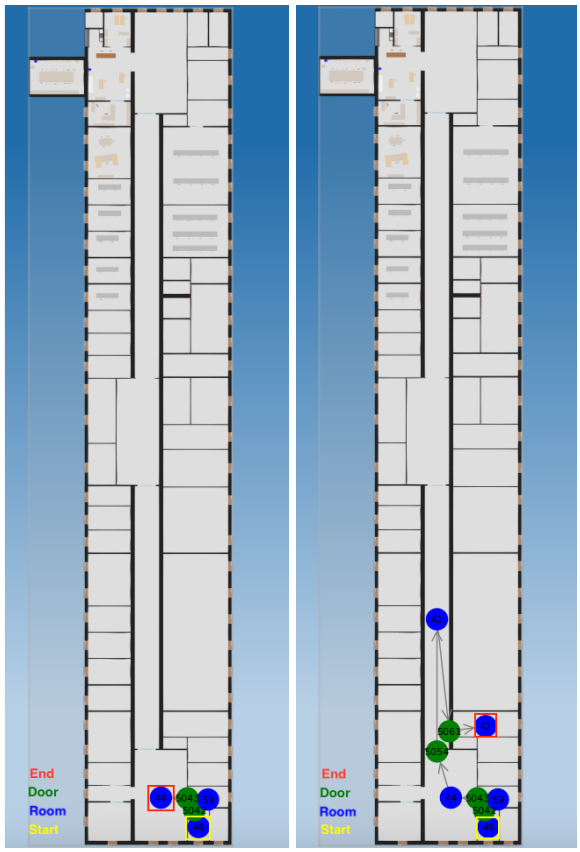
In contrast, the o1-preview model demonstrates stronger spatial and dimensional reasoning. It suggests scanning positions that are well-aligned with the room's dimensions, ensuring broader coverage of the target surfaces. For example, as illustrated in fig. 4.3(d),

the o1-preview model's scanning orientation, while not perfectly optimal, manages to capture a larger portion of the surface compared to GPT-4o. This highlights the o1-preview model's ability to prioritize maximizing coverage even when the exact scanning angle deviates slightly from the ideal. However, in the case of wall scanning with the same surface area, the o1-preview model did not propose a position that ensures maximum coverage. Instead, it suggested an optimal orientation that achieves the highest possible coverage within the constraints of that orientation.

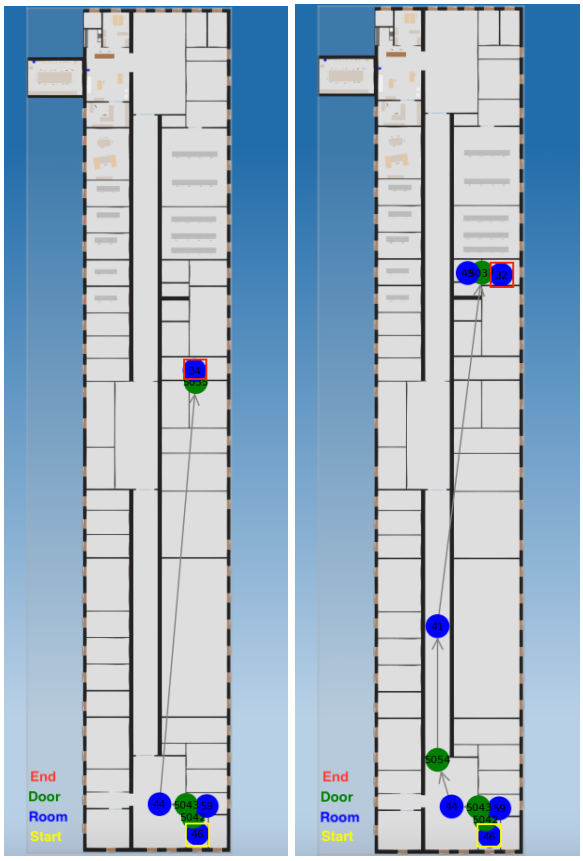
When scanning building components like doors and windows, both models occasionally display hallucinations or generate suboptimal plans. For instance, in scenarios involving doors that connect two rooms, GPT-4o often proposes partially feasible plans that lack precision. The o1-preview model, while generally more accurate, sometimes suggests scanning from an adjacent room instead of the designated one. This happened even though the prompt explicitly instructed the model to propose a scanning position within the room containing the defect.

Similarly, when tasked with scanning windows, GPT-4o's plans tend to be limited in scope, frequently neglecting corners or edges. The o1-preview model performs significantly better in this regard, producing near-optimal scanning plans that capture the majority of the window area. However, in some cases, small portions, particularly in corner areas, remain unscanned, indicating room for further improvement even in the stronger model.

Overall, the comparison highlights the strengths of the o1-preview model in generating more comprehensive scanning plans for both primary and secondary building components. However, both models still encounter occasional challenges, particularly when handling objects that span multiple spaces or require precise boundary recognition.

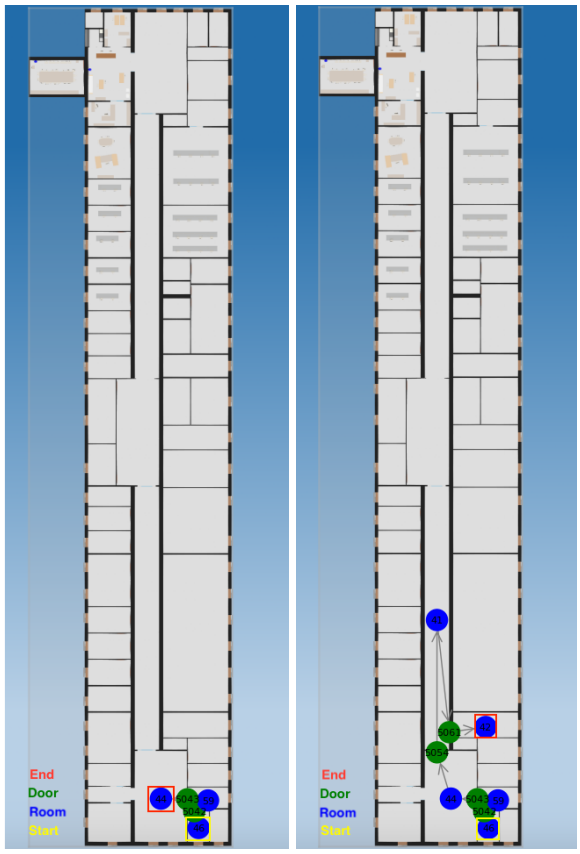


(a) node distance 2 (b) node distance 4

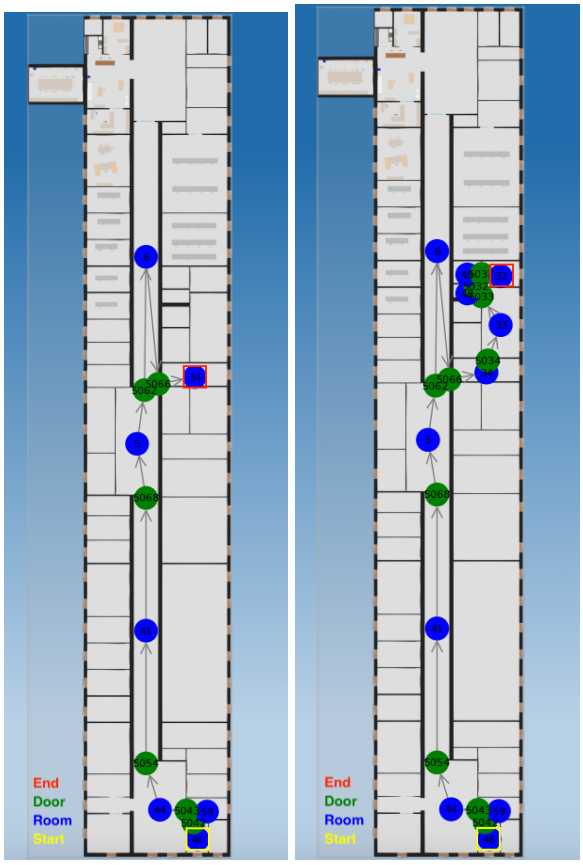


(c) node distance 6 (d) node distance 10

Figure 4.1: The paths generated by GPT-4o with different node distances

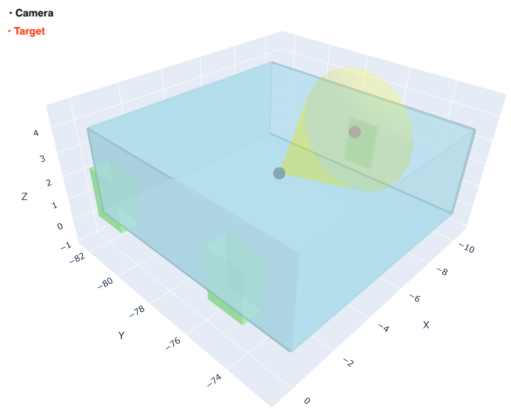


(a) node distance 2 (b) node distance 4

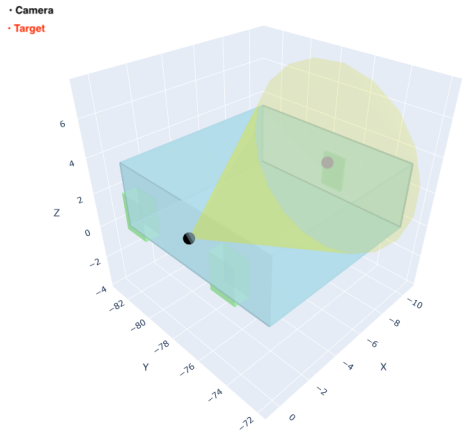


(c) node distance 6 (d) node distance 10

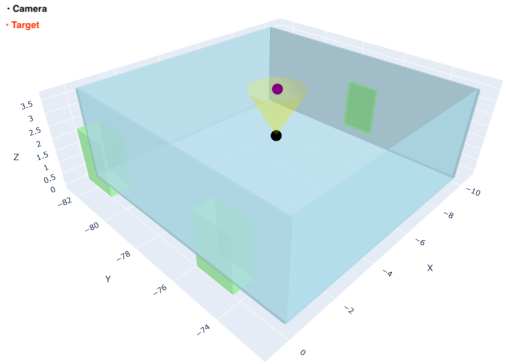
Figure 4.2: The paths generated by o1-preview with different node distances



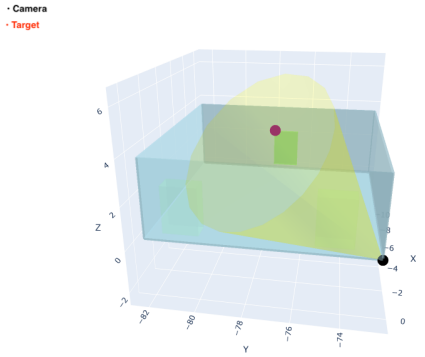
(a) scanning plan for a wall by GPT-4o



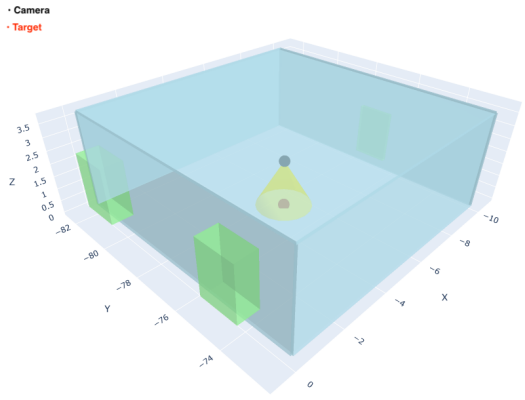
(b) scanning plan for a wall by o1-preview



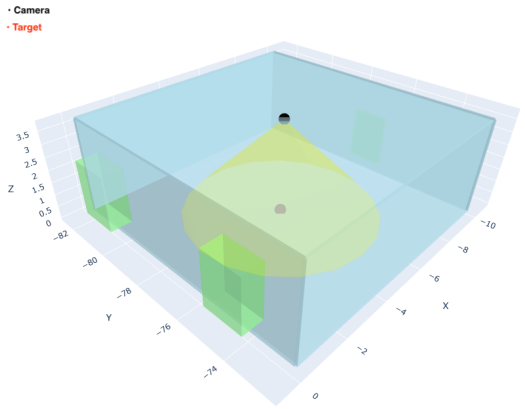
(c) scanning plan for a ceiling by GPT-4o



(d) scanning plan for a ceiling by o1-preview

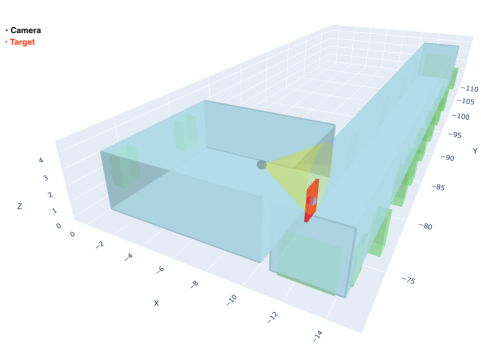


(e) scanning plan for a floor by GPT-4o

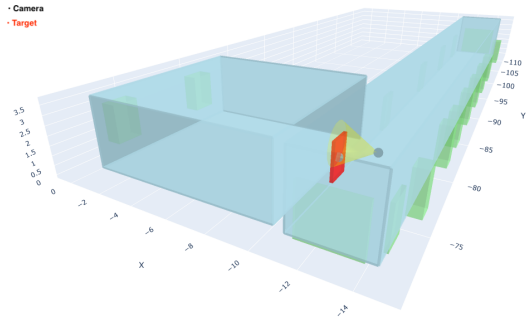


(f) scanning plan for a floor by o1-preview

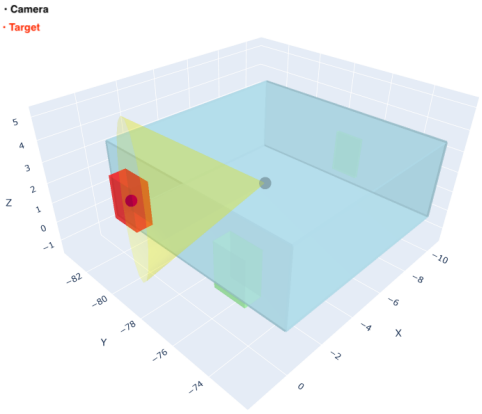
Figure 4.3: Comparison of the Scanning Plans between GPT-4o and o1-preview (1)



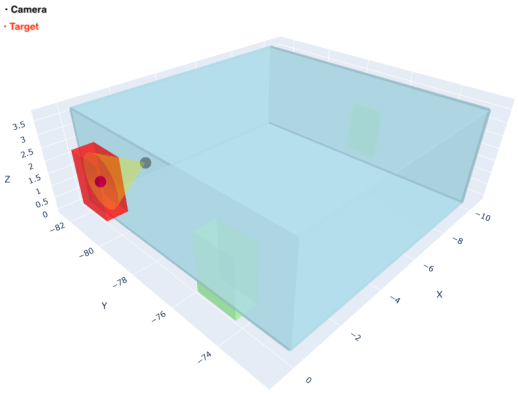
(a) scanning plan for a door by GPT-4o



(b) scanning plan for a door by o1-preview



(c) scanning plan for a window by GPT-4o



(d) scanning plan for a window by o1-preview

Figure 4.4: Comparsion of the Scanning Plans between GPT-4o and o1-preview (2)

4.4 Summary of Finding

The two models evaluated in this study, GPT-4o and o1-preview, demonstrated distinct performance patterns across the tested modules. As shown in table 4.1 and table 4.2, the GPT-4o model exhibited strong accuracy in simple node searches but struggled with queries requiring complex spatial reasoning or multiple attribute considerations, sometimes failing at the first attempt or achieving only partial success. It also encountered difficulties in selecting robots based on cost criteria and occasionally omitted essential steps when planning tasks. In contrast, the o1-preview model achieved successful outcomes on the first attempt for the same tasks and proposed more accurate plans, particularly for longer navigation paths and advanced spatial arrangements.

As indicated in table 4.5, GPT-4o could generate optimal routes for short distances but often skipped intermediate nodes for longer paths involving six or more doors. Similarly, while it correctly identified orientations for scanning tasks involving basic elements like walls, ceilings, and floors, it often did not achieve full coverage of the target surfaces. The o1-preview model, on the other hand, provided more extensive scanning coverage and handled complex spatial queries better. However, it did not always produce optimal solutions for floors or doors and occasionally proposed scanning positions outside the specified room constraints.

Overall, GPT-4o delivered a strong performance for simple tasks and shorter paths but showed limitations when confronted with more complex requests. The o1-preview model demonstrated superior reasoning for intricate scenarios, suggesting more comprehensive scanning areas and effectively managing longer routes. Yet, some conditions remained unmet, and certain tasks were only partially addressed, indicating potential areas for further refinement.

Chapter 5

Discussion

This section synthesizes the findings from evaluating the LLM-based modules, including semantic search, navigation and scanning planning, and robot task planning, within the proposed framework using BIM3DSG. The results highlight both the potential and the current limitations of applying LLMs to complex spatial and dimensional reasoning tasks in the AEC domain. While GPT-4o performed well in simpler semantic searches and planning tasks, it struggled with more advanced spatial reasoning, detailed dimensional analysis, and generating complete scanning plans for planar objects. In comparison, the o1-preview model handled complex queries, navigation tasks, and certain scanning conditions with greater accuracy. However, some of the scanning plans it produced were only partially correct or incomplete.

These findings suggest that LLMs offer a promising starting point for automation and could potentially replace or assist algorithms like A* or complex database queries in specific scenarios. However, they are not yet fully optimized and are prone to hallucinations. The following subsections provide a closer look at these observations. I begin by revisiting the navigation and scanning planners to examine the strengths and weaknesses of LLMs in spatial and dimensional reasoning. This is followed by a discussion on the robot task planner, focusing on how LLMs apply general and domain-specific knowledge to reduce the need for human intervention. Lastly, I review the role of the BIM3DSG data format in efficiently encoding building information and discuss future directions while connecting the findings to the key research questions of this study.

5.1 Analysis of LLM-based Modules

5.1.1 Semantic Searcher and Robot Selector

The proposed framework employs LLMs for semantic search and robot selection tasks. Unlike traditional database queries that rely on predefined logic and complex post-processing, this approach allows the LLM to infer the appropriate nodes based on natural language requests. The LLM can identify relevant data through its internal reasoning, even when complex spatial reasoning or partial attributes must be understood. Notably, while the GPT-4o model struggled with some spatial reasoning tasks, the o1-preview model demonstrated that it could interpret user requests requiring sophisticated spatial relationships and retrieve relevant data. This result suggests that LLMs can partially replace traditional databases for complex semantic queries, although issues such as hallucination, token limits, and increased response times and costs remain practical constraints. Future work

could integrate LLMs with database systems in a hybrid manner, allowing LLMs to generate database queries and offload the actual data retrieval to a database engine for greater efficiency and scalability.

5.1.2 Navigation and Scanning Planners

The navigation planner uses spatial relationships and dimensional reasoning to determine optimal paths. The planner with GPT-4o successfully identified paths involving four to five node distances but failed when planning routes that required passing through more than six doors. In contrast, the o1-preview model successfully planned all tested distances, showing superior spatial reasoning. This suggests that LLM-based navigation planning could potentially replace graph-searching algorithms for path planning in certain scenarios.

In the scanning planner module, both GPT-4o and o1-preview faced challenges in generating fully optimal plans. While they could align scanning positions at central points of building surfaces (e.g., walls, ceilings, floors) to cover defined areas without distortion, neither model consistently identified the best positions or orientations for complete coverage. More complex tasks, such as interpreting FOV, identifying defect target areas, and managing intricate coordinates, proved difficult. Although o1-preview showed slightly better performance, it still produced hallucinations, particularly for large surfaces that required multiple scanning positions.

These results suggest that the unclear prompt guidance on optimal position selection led to hallucination, especially when a single position could not cover large areas. When a single camera position cannot cover the entire area, the prompt must specify whether to prioritize maximum coverage with potential distortion or limited coverage without it. Future research should focus on refining prompt instructions to address these ambiguous prompts and explore hybrid solutions that combine LLMs with computer vision and spatial analysis to improve scanning accuracy and efficiency.

5.1.3 Robot Task Planner

The robot task planner module generates rational action sequences based on a given robot configuration and user requests. Even with relatively simple actions tested in this study, the LLM showed the ability to apply both domain-specific knowledge and general reasoning to propose feasible action plans. This demonstrates that LLMs, without complex domain-specific programming, can assist in automating robot decision-making. Such capabilities are promising for tasks like corrective maintenance, where rapid and informed decision-making can reduce human intervention. Although the current scope is limited to a small set of actions and does not address low-level robot control, future work could expand to more complex tasks and integrate with actual robots to execute the proposed actions.

5.1.4 BIM3DSG

The proposed BIM3DSG data structure bridges the gap between BIMSegGraphs and conventional 3D Scene Graphs, enabling the integration of BIM data into robotic applications within the AEC domain. it demonstrates an effective solution and provides a crucial data structure for feeding building information into LLMs. This structure effectively maps the flat segmentation data from BIMSegGraphs into an expanded 3D scene graph format, tailored for robotic applications in building maintenance.

To handle diverse building maintenance tasks, such as semantic searching, robot selection, task planning, navigation planning, and scanning planning, the data structure organizes nodes and links to serve distinct purposes. Nodes manage only hierarchical information about building objects, while links represent only room connectivity for navigation. This separation helps overcome LLM token limitations without needing additional algorithms. This approach effectively supports large environments with up to 62 rooms in 3D scene graphs.

However, the current representation is constrained by the assumption of rectangular object shapes, limiting its ability to capture intricate geometries. Future research should focus on developing more generalized 3D scene graph representations capable of describing complex and non-rectangular building elements to further enhance the flexibility and accuracy of this approach.

5.2 Addressing Key Research Questions

This study aims to fill notable research gaps in applying robotics to corrective maintenance tasks in the AEC domain. Existing automated approaches generally focus on preventive maintenance, leaving unpredictable corrective tasks underexplored. Additionally, while 3D Scene Graphs offer a novel solution for managing large building environments in robotics, they omit essential building structures in their data representation. On the other hand, BIMSegGraphs have laid the groundwork for mapping volumetric BIM data into room-segmented formats but have not been fully integrated with robotic applications. Finally, research on how well LLMs can perform in the AEC domain, particularly in spatial and dimensional reasoning, remains limited. To tackle these gaps, three core research questions frame the scope of this work:

5.2.1 RQ1: Automated Robot Planning for Corrective Maintenance using LLMs

This study explores how LLMs can serve as a powerful high-level decision-maker in robotics for the AEC domain, bridging existing gaps between the robotic library and AEC workflows. Specifically, five LLM-based modules are proposed, each capable of interpreting diverse

user requests and making various decisions that enable quick, responsive corrective maintenance with minimal human intervention.

While the base model (GPT-4o) displays limited spatial and dimensional reasoning, it efficiently handles simpler tasks such as node searches, navigation planning, scanning strategies, robot selection, and task planning. However, it occasionally hallucinates in more complex scenarios. In contrast, the o1-preview model, a more advanced reasoning model from OpenAI, demonstrates remarkable performance that can potentially replace traditional algorithms for complex tasks. While this framework has been validated in only a few limited scenarios, it provides an initial foundation for developing automated robotic applications in building corrective maintenance by leveraging LLMs and BIM data.

5.2.2 RQ2: BIM-driven 3D Scene Graphs (BIM3DSG)

In this study, I introduce the BIM3DSG format as a unified 3D scene graph structure that preserves core architectural details and integrates the room-segmented BIM data from BIMSegGraphs. This approach enables direct use of BIM data in robotics for building maintenance tasks. The pipeline expands segmented BIM graphs, like BIMSegGraphs, into a 3D scene graph format that aligns with the needs of the AEC domain, allowing room-wise segmented BIM data to be integrated with robotics.

A key challenge in LLM-based robotic planning arises from the token limitations associated with representing expansive real-world building data, which is often too large to be passed directly to LLMs. To address this, BIM3DSG separates hierarchical node information from space connectivity data. This modular design allows each LLM-based module to access only the relevant subset of the environment's data. Consequently, even large environments containing up to 62 rooms can be effectively managed without resorting to external algorithms or additional data processing steps.

In demonstrating the feasibility of this approach, BIM3DSG emerges as a potential standard data structure for mapping BIM-driven building information to robotics applications. By providing a flexible yet comprehensive representation of both semantic and spatial relationships, this framework lays the groundwork for advanced corrective maintenance tasks in large-scale buildings.

5.2.3 RQ3: Integration of LLM-base Task Planning with a Robotic Library for building maintenance

In this research, I demonstrate how LLMs can be integrated into a robotics library to generate actionable instructions such as navigation paths, scanning trajectories, and action sequences, that can be executed as real robot commands for building maintenance tasks. By linking the robotics library with BIM3DSG, the proposed framework creates an automated end-to-end pipeline that generates high-level robot plans, enabling robots to perform diverse building maintenance tasks with minimal human intervention.

This pipeline also clarifies which semantic and dimensional attributes of a robot (e.g., maximum reachable height, FOV) are crucial for LLM-based task planning in building maintenance. For example, I outline five fundamental corrective maintenance use cases that a one-arm robot can perform: (1) cleaning surfaces, (2) painting surfaces, (3) disinfecting elements, (4) filling cracks, and (5) picking up trash. I define the high-level actions required for each task and provide an evaluation. These efforts lay the groundwork for more advanced robotic applications in building maintenance. Furthermore, the suggested LLM-based robot path planning approach leverages BIM data by focusing on door-room sequences instead of the conventional scanning-position method found in conventional 3D scene graphs (Rana et al., 2023). This strategy makes it easier to integrate BIM-driven data for robotic path planning.

Overall, the proposed framework delivers an end-to-end solution from defect identification and robot selection to navigation, scanning, and task planning, showing the potential of combining BIM data with LLM-based methodologies for building maintenance tasks.

5.3 Future Directions

This initial study highlights the potential of LLMs in handling various AEC tasks—semantic searching, navigation, scanning, and robot task planning. While GPT-4o performed well on simpler tasks, the o1-preview model successfully addressed more complex challenges, demonstrating the value of advanced LLM reasoning for spatial and dimensional tasks. Moving forward, integrating LLMs with database engines could create a hybrid approach that leverages LLM inference for complex semantic queries and delegates stable, scalable retrieval to databases. Additionally, incorporating computer vision and robotics hardware could enable fully automated corrective maintenance procedures. These directions offer promising avenues for expanding LLM applications in AEC domains, improving both efficiency and adaptability.

Chapter 6

Conclusion

This study proposes a novel approach that integrates robotics with LLMs to address unpredictable corrective maintenance tasks in the AEC domain. While previous research on automating building maintenance with robots has primarily focused on preventive maintenance due to its predictable nature, corrective maintenance demands rapid responses to unexpected failures. To handle such dynamic scenarios, robots require automated high-level decision-making. To this end, I presented (1) an end-to-end LLM-based planner comprising a semantic searcher, robot selector, navigation planner, scanning planner, and robot task planner; (2) a BIM-driven 3D Scene Graph structure called BIM3DSG, and (3) a strategy to integrate these components with a robotics library for potential robotic execution.

First, the experiments demonstrated that LLM-based modules can automate the essential decision-making stages for corrective maintenance. For relatively simpler tasks such as node searches, short path planning, and basic scanning position proposals, GPT-4o performed adequately. In contrast, more complex tasks that rely heavily on spatial and dimensional reasoning (e.g., long-route navigation or complex semantic searching) benefited greatly from the advanced reasoning capabilities of the o1-preview model. This indicates that LLMs can replace or complement traditional algorithms for robot task planning, such as A* and complex database queries, by interpreting user instructions in natural language without additional training costs. Nonetheless, challenges persist, including hallucinations, token limitations, and increased response times and costs for complex tasks.

Next, The newly proposed BIM3DSG structure introduces a novel mapping approach that combines room-segmented BIM data from BIMSegGraphs with the hierarchical organization of conventional 3D Scene Graphs, creating an enhanced 3D representation for robotic applications in building maintenance. By managing spatial data separately in the node and link layers, BIM3DSG effectively delivers the necessary spatial information to each LLM-based module. This approach enables the representation of large environments without exceeding token limits or requiring additional algorithms. Consequently, BIM3DSG shows promise as a standard data structure, bridging segmented BIM data with robotics to enable LLM-based robot task planning for corrective maintenance operations.

Finally, I demonstrated how key robot configurations (e.g., reach, FOV, and available action sets) relevant to building maintenance can be utilized by LLMs to enhance spatial and dimensional reasoning, generating feasible action sequences. By testing a limited set of corrective maintenance tasks (cleaning, painting, disinfecting, crack filling, and trash collection), the results indicate how LLM can be utilized to integrate BIM data with the

robotic library to handle diverse building maintenance use cases. This research effectively narrows the gap between robotics and BIM, pointing to the possibility of reducing human intervention and achieving rapid, autonomous corrective maintenance.

In future work, several directions merit exploration. First, hybrid frameworks that integrate LLMs with database engines could yield accurate high-level inferences while maintaining scalable data retrieval. Second, extending the research to demonstrate low-level robot execution by combining computer vision with robotics hardware could enable real-time detection of defects or changes in building conditions, surpassing basic scan planning. Third, BIM3DSG can be extended to accommodate more complex and irregular geometries, further broadening its applicability.

Appendix A

Code Examples

code A.1: An Example Node of BIM-to-Room Result in graphgml

```
{
    "Unnamed: 0": 0.0,
    "cp_x": -8.5624,
    "cp_y": -16.8071,
    "cp_z": 1.013,
    "extent_pca1": 2.026,
    "extent_pca2": 1.052,
    "extent_pca3": 0.53,
    "pca1x": -0.0,
    "pca1y": 0.0,
    "pca1z": -1.0,
    "pca2x": -1.0,
    "pca2y": 0.0,
    "pca2z": 0.0,
    "pca3x": 0.0,
    "pca3y": 1.0,
    "pca3z": 0.0,
    "normal_x": 0.0,
    "normal_y": 1.0,
    "normal_z": 0.0,
    "ifc_guid": "0VHNU8CzX2rfsiwnl8p4GF",
    "node_type": "element",
    "label": 4,
    "color": "255,192,0,1.0",
    "discipline": 1,
    "room": "[307]",
    "id": "14"
},
```

code A.2: Full Action Specification in Python

```
[
  {
    "action": "loadArm",
    "parameters": ["equipment", "material=None"],
    "preconditions": [
      "if equipment == 'sprayGun' and material != None"
    ]
  },
  {
```

```
  "action": "unloadArm",
  "parameters": []
},
{
  "action": "collectTrashes",
  "parameters": [],
  "preconditions": [
    "equipment == 'gripper'",
    "action == 'scanTrashes'"
  ]
},
{
  "action": "scanTrashes",
  "parameters": ["surface_id"],
  "preconditions": [
    "equipment == None",
    "element_type == 'floor'"
  ]
},
{
  "action": "placeInTrashBag",
  "parameters": [],
  "preconditions": [
    "equipment == 'gripper'",
    "action == 'collectTrashes'"
  ]
},
{
  "action": "spraySurface",
  "parameters": ["surface_id"],
  "preconditions": [
    "equipment == 'sprayGun'",
    "material != None"
  ]
},
{
  "action": "wipeSurface",
  "parameters": ["surface_id"],
  "preconditions": [
    "equipment == 'wiper'",
    "surfaceIsWet == true"
  ]
},
{
  "action": "spreadPaint",
  "parameters": ["surface_id"],
  "preconditions": [
    "equipment == 'roller'",
    "previous_action == 'spraySurface'",
```

```
      "material␣==␣'paint'"
    ]
  },
  {
    "action": "spreadFiller",
    "parameters": ["surface_id"],
    "preconditions": [
      "equipment␣==␣'scrapper'",
      "previous_action␣==␣'spraySurface'",
      "material␣==␣'filler'"
    ]
  }
]
```

code A.3: An Example Nodes layer in BIM3DSG

```
"nodes": {
        "spaces": [
            {
                "id": 0,
                "surfaces": [
                    1069,
                    2001,
                    1078,
                    1092,
                    3041,
                    1186
                ],
                "location": [
                    -8.0035,
                    -49.4861,
                    7.85
                ],
                "size": [
                    4.11,
                    5.165,
                    3.7
                ]
            }, ...]
    "surfaces": [
                {
                    "id": 1000,
                    "type": "wall",
                    "components": [
                        5051
                    ],
                    "location": [
                        -18.18925,
                        -120.4114,
                        1.85
```

```
                    ],
                    "size": [
                        6.381,
                        0.0,
                        3.7
                    ]
                },
                ...]

    "components": [
                {
                    "id": 5000,
                    "type": "door",
                    "location": [
                        -14.6785,
                        -19.5285,
                        1.013
                    ],
                    "size": [
                        0.69,
                        1.252,
                        2.026
                    ]
                },
                ...]
```

code A.4: An Example Link layer in BIM3DSG

```
"links": [
        {
            "spaces": [
                9,
                6
            ],
            "via": 5000
        },
        {
            "spaces": [
                8,
                9
            ],
            "via": 5001
        },
        ...
        ]
```

Bibliography

Armeni, I., He, Z.-Y., Gwak, J., Zamir, A. R., Fischer, M., Malik, J., & Savarese, S. (2019). 3d scene graph: A structure for unified semantics, 3d space, and camera. *Proceedings of the IEEE International Conference on Computer Vision*, 5664–5673.

Bae, J., Shin, D., Ko, K., et al. (2023). A survey on 3d scene graphs: Definition, generation and application. In *Robot intelligence technology and applications 7*. Springer.

Brohan, A., Chebotar, Y., Finn, C., Hausman, K., Herzog, A., Ho, D., Ibarz, J., Irpan, A., Jang, E., Julian, R., et al. (2023). Do as i can, not as i say: Grounding language in robotic affordances. *Conference on Robot Learning*, 287–318.

Brunete, A., Hernando, M., Torres, J. E., & Gambao, E. (2012). Heterogeneous multi-configurable chained microrobot for the exploration of small cavities. *Automation in Construction*, *21*, 184–198. https://doi.org/10.1016/j.autcon.2011.05.013

Buruzs, A., Šipetić, M., Blank-Landeshammer, B., & Zucker, G. (2022). Ifc bim model enrichment with space function information using graph neural networks. *Energies*, *15*, 2937. https://doi.org/10.3390/en15082937

Chang, A., Dai, A., Funkhouser, T., Halber, M., Nießner, M., Savva, M., Song, S., Zeng, A., & Zhang, Y. (2017). Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*.

Chen, B., Xu, Z., Kirmani, S., Ichter, B., Driess, D., Florence, P., Sadigh, D., Guibas, L., & Xia, F. (2024). Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. *arXiv preprint*. http://arxiv.org/abs/2401.12168

Chen, C., & Tang, L. (2019). Bim-based integrated management workflow design for schedule and cost planning of building fabric maintenance. *Automation in Construction*, *107*, 102944. https://doi.org/10.1016/j.autcon.2019.102944

Cheng, Y., Jiang, F., Han, Z., Wang, H., Zhou, F., Li, Z., Wang, B., & Huang, Y. (2024). Robot navigation based on 3d scene graphs with the llm tooling. *Proceedings of the 2024 WRC Symposium on Advanced Robotics and Automation (WRC SARA)*. https://doi.org/10.1109/WRCSARA64167.2024.10685831

Claire, F. (2024). Bim-room-graph framework github repository [Accessed: 2024-12-22]. %5Curl%7Bhttps://github.com/fclairec/bim_room_graph%7D

Driess, D., Xia, F., Sajjadi, M. S. M., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., Huang, W., Chebotar, Y., Sermanet, P., Duckworth, D., Levine, S., Vanhoucke, V., Hausman, K., Toussaint, M., Greff, K., ... Florence, P. (2023). Palm-e: An embodied multimodal language model [arXiv preprint]. https://arxiv.org/abs/2303.03378

Gan, V. (2022). Bim-based graph data model for automatic generative design of modular buildings. *Autom. Constr.*, *134*, 104062. https://doi.org/10.1016/j.autcon.2021.104062

Halder, S., & Afsari, K. (2023). Robots in inspection and monitoring of buildings and infrastructure: A systematic review. *Applied Sciences*, *13*(4), 2304. https://doi.org/10.3390/app13042304

Hamledari, H., Sajedi, S., McCabe, B., & Fischer, M. (2020). Automation of inspection mission planning using 4d bims and in support of unmanned aerial vehicle–based data collection [Published online: December 23, 2020]. *Journal of Construction Engineering and Management*, *146*(12), 04020130. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001936

Han, D., McInroe, T., Jelley, A., Albrecht, S. V., Bell, P., & Storkey, A. (2024). LLM-Personalize: Aligning LLM planners with human preferences via reinforced self-training for housekeeping robots [https://arxiv.org/abs/2404.14285].

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., et al. (2022). Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*.

Kannan, S. S., Venkatesh, V. L. N., & Min, B.-C. (2024). Smart-llm: Smart multi-agent robot task planning using large language models. https://arxiv.org/abs/2309.10062

Khalili, A., & Chua, D. (2015). Ifc-based graph data model for topological queries on building elements. *J. Comput. Civ. Eng.*, *29*, 04014046. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000331

Liu, B., Jiang, Y., Zhang, X., Liu, Q., Zhang, S., Biswas, J., & Stone, P. (2023). Llm+p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.

Liu, L., Wang, X., Yang, X., Liu, H., Li, J., & Wang, P. (2023). Path planning techniques for mobile robots: Review and prospect. *Expert Systems with Applications*, *227*, 120254.

López, J., Pérez, D., Paz, E., & Santana, A. (2013). Watchbot: A building maintenance and surveillance system based on autonomous robots. *Robotics and Autonomous Systems*, *61*(12), 1559–1571. https://doi.org/10.1016/j.robot.2013.06.012

Marocco, M., & Garofolo, I. (2021). Integrating disruptive technologies with facilities management: A literature review and future research directions. *Automation in Construction*, *131*, 103917. https://doi.org/10.1016/j.autcon.2021.103917

Meng, S., Wang, Y., Yang, C.-F., Peng, N., & Chang, K.-W. (2024). Llm-a*: Large language model enhanced incremental heuristic search on path planning. *arXiv preprint arXiv:2407.02511*.

Mo, Y., & et al. (2020). Automated staff assignment for building maintenance using natural language processing. *Automation in Construction*, *113*, 103150. https://doi.org/10.1016/j.autcon.2020.103150

Moon, S.-M., Huh, J., Hong, D., Lee, S., & Han, C.-S. (2015). Vertical motion control of building façade maintenance robot with built-in guide rail. *Robotics and Computer-Integrated Manufacturing*, *31*, 11–20. https://doi.org/10.1016/j.rcim.2014.06.006

OpenAI. (2024). *Reasoning guide for openai platform* [Accessed: 2024-12-11]. https://platform.openai.com/docs/guides/reasoning

Park, J. W., Chen, J., & Cho, Y. K. (2021). Self-corrective knowledge-based digital twin system for remote control of smart factories. *Smart and Sustainable Manufacturing Systems*, *5*(1).

Rana, K., Haviland, J., Garg, S., Abou-Chakra, J., Reid, I., & Underhauf, N. S. (2023). Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning. *7th Conference on Robot Learning (CoRL 2023)*.

Silver, T., Hariprasad, V., Shuttleworth, R. S., Kumar, N., Lozano-Pérez, T., & Kaelbling, L. P. (2022). Pddl planning with pretrained large language models. *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

Song, C. H., Wu, J., Washington, C., Sadler, B. M., Chao, W.-L., & Su, Y. (2023). Llm-planner: Few-shot grounded planning for embodied agents with large language models [ICCV 2023]. *arXiv preprint arXiv:2212.04088*. https://doi.org/10.48550/arXiv.2212.04088

Tauscher, E., Bargstädt, H.-J., & Smarsly, K. (2016). Generic bim queries based on the ifc object model using graph theory. *Proceedings of the 16th International Conference on Computing in Civil and Building Engineering (ICCCBE)*, 1–8. https://www.researchgate.net/publication/296602879_Generic_BIM_queries_based_on_the_IFC_object_model_using_graph_theory

Torok, M. M., Golparvar-Fard, M., & Kochersberger, K. B. (2014). Image-based automated 3d crack detection for post-disaster building assessment. *Journal of Computing in Civil Engineering*, *28*(5), A4014004. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000334

Valero, E., Forster, A., Bosché, F., Hyslop, E., Wilson, L., & Turmel, A. (2019). Automated defect detection and classification in ashlar masonry walls using machine learning. *Automation in Construction*, *106*, 102846. https://doi.org/10.1016/j.autcon.2019.102846

Vilgertshofer, S., & Borrmann, A. (2017). Using graph rewriting methods for the semi-automatic generation of parametric infrastructure models. *Adv. Eng. Inform.*, *33*, 502–515. https://doi.org/10.1016/j.aei.2017.05.003

Xia, F., Zamir, A. R., He, Z., Sax, A., Malik, J., & Savarese, S. (2018). Gibson env: Real-world perception for embodied agents. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9068–9079.

Zhao, Q., Li, Y., Hei, X., & Yang, M. (2020). A graph-based method for ifc data merging. *Adv. Civ. Eng.*, *2020*, 8782740. https://doi.org/10.1155/2020/8782740

Zheng, J., Wang, B., Li, L., Lee, Z., Schiebener, D., Croft, E., Gupta, S., & Fox, D. (2023). Grid: Scene-graph-based instruction-driven robotic task planning. https://arxiv.org/abs/2309.07726

Zhu, J., Chong, H.-Y., Zhao, H., Wu, J., Tan, Y., & Xu, H. (2022). The application of graph in bim/gis integration (M. Shafique, Ed.) [Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.]. *Buildings*, *12*(12), 2162. https://doi.org/10.3390/buildings12122162