

Article

Robot-Enabled Construction Assembly with Automated Sequence Planning Based on ChatGPT: RoboGPT

Hengxu You, Yang Ye, Tianyu Zhou, Qi Zhu and Jing Du * 

Engineering School of Sustainable Infrastructure & Environment, University of Florida, Gainesville, FL 32611, USA; you.h@ufl.edu (H.Y.); ye.yang@ufl.edu (Y.Y.); zhoutianyu@ufl.edu (T.Z.); qi.zhu@nist.gov (Q.Z.)

* Correspondence: eric.du@essie.ufl.edu

Abstract: Robot-based assembly in construction has emerged as a promising solution to address numerous challenges such as increasing costs, labor shortages, and the demand for safe and efficient construction processes. One of the main obstacles in realizing the full potential of these robotic systems is the need for effective and efficient sequence planning for construction tasks. Current approaches, including mathematical and heuristic techniques or machine learning methods, face limitations in their adaptability and scalability to dynamic construction environments. To expand the current robot system's sequential understanding ability, this paper introduces RoboGPT, a novel system that leverages the advanced reasoning capabilities of ChatGPT, a large language model, for automated sequence planning in robot-based assembly applied to construction tasks. The proposed system adapts ChatGPT for construction sequence planning and demonstrates its feasibility and effectiveness through experimental evaluation including two case studies and 80 trials involving real construction tasks. The results show that RoboGPT-driven robots can handle complex construction operations and adapt to changes on the fly. This paper contributes to the ongoing efforts to enhance the capabilities and performance of robot-based assembly systems in the construction industry, and it paves the way for further integration of large language model technologies in the field of construction robotics.



Citation: You, H.; Ye, Y.; Zhou, T.; Zhu, Q.; Du, J. Robot-Enabled Construction Assembly with Automated Sequence Planning Based on ChatGPT: RoboGPT. *Buildings* **2023**, *13*, 1772. <https://doi.org/10.3390/buildings13071772>

Academic Editors: Shuai Han and Yantao Yu

Received: 8 June 2023
Revised: 6 July 2023
Accepted: 10 July 2023
Published: 12 July 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: robot-assembly; sequential learning; ChatGPT

1. Introduction

Robot-based construction assembly refers to the use of robotic systems for joining together various building components, materials, and systems to form a complete structure or a part of a structure [1]. It has emerged as a promising solution to address various challenges including increasing costs, labor shortages, project schedules, and the increasing demand for safe and efficient construction processes [2]. The use of robotic systems and the corresponding changes to the existing construction workflow are expected to significantly enhance productivity, reduce construction costs, and improve safety of construction projects [3]. Moreover, robot-based assembly systems can perform construction tasks that are repetitive, hazardous, or require high precision, thereby alleviating the burden on human workers [4].

Despite the potential benefits of robot-based assembly in construction, one of the main challenges faced by these systems is the need for effective and efficient sequence planning. A construction task often consists of a variety of interdependent steps that must be executed in a specific sequence order [5]. For example, installing a plumbing system requires a proper sequence of connecting pipes of different diameters and lengths, and necessitates the use of the appropriate couplings. Similarly, bricklaying requires placing the right bricks in the corresponding locations in the correct sequence. Many of these sequence planning tasks rely on spontaneous decisions, as construction tasks are often less

predictable and difficult to plan out due to varying site conditions, resource availability, and evolving requirements [6]. As a result, construction workers often need to perform manual sequence planning on the fly, which involves determining the optimal order of construction steps and taking into account the corresponding logistic considerations. Manual sequence planning is a time-consuming and labor-intensive process, requiring a significant amount of experience to ensure quality and accuracy. Moreover, the complexity of construction task sequences can vary significantly depending on the specific construction project, further increasing the difficulty of the task [7]. Without an effective method for automated sequence planning, robot-based construction automation would not be scalable for meeting the needs of real-world complex construction tasks.

In order to enable automation systems (including construction robotics) to handle more complex multi-step operational tasks, efforts have been made to explore heuristic-based methods or learning-based methods. Early investigations included the use of mathematical and heuristic techniques in tackling the complex problem of sequence planning, such as mixed-integer linear programming (MILP) (e.g., [8]). Recently, advances in machine learning have been leveraged to support complex sequence planning with various constraints (e.g., [9]). These techniques aim to optimize operational sequences by considering factors such as precedence constraints, resource availability, and task interdependencies. By integrating these approaches with robotic systems, researchers expect to develop more efficient and adaptable solutions that can manage the inherent complexities and uncertainties of construction operations.

However, these methods have certain limitations that hinder their effectiveness in addressing the dynamic nature of construction projects. On the one hand, mathematical and heuristic techniques often involve the development of tailored algorithms (by human experts) that leverage domain-specific knowledge and rules [10]. While these methods can effectively navigate the complex solution space for complex and variable construction tasks, they may impose a significant computational overhead due to the need for continuous adaptation and refinement of the heuristics as the construction process evolves. On the other hand, although machine learning methods, such as genetic algorithms and neural networks, can adapt to dynamic scenarios much easier compared to mathematical and heuristic techniques, they require a significant amount of training data to achieve accurate results [11]. In construction operations, where site conditions and project requirements can change frequently, acquiring sufficient training data for every possible scenario is challenging, limiting the adaptability of these methods to dynamic environments.

The primary objective of this research is to design and evaluate a new system, named RoboGPT, that leverages the capabilities of ChatGPT to achieve automated sequence planning in robotic assembly for construction tasks. ChatGPT, as an advanced large language model (LLM), has demonstrated remarkable capabilities in understanding and generating human-like text, which relies on a reasoning ability for understanding the inherent structures of a sequence [12]. By integrating ChatGPT into the construction process, we aim to minimize the reliance on manual intervention, reduce planning time, and increase the overall efficiency of robot-based assembly systems in the construction industry. Specifically, in this paper we will show how we adapted ChatGPT for the purpose of automated sequence planning in robot-based assembly for construction applications and demonstrate the feasibility and effectiveness of the proposed approach through an experimental evaluation, including comparing the ability of ChatGPT-driven robots in handling complex construction operations and adapting to changes on the fly. By accomplishing these goals, this paper will contribute to the ongoing efforts to enhance the capabilities and performance of robot-based assembly systems in the construction industry and pave the way for further integration of LLM technologies in the field of construction robotics.

The main hypothesis of this paper is that the reasoning capabilities inherent in ChatGPT can be harnessed to create a flexible and efficient sequence planning system, RoboGPT, for construction tasks. To validate the flexibility of the proposed system, we assess its performance across two distinct tasks in varying fields and offer a qualitative analysis of

the outcomes. For determining the effectiveness of RoboGPT, we undertake an extensive evaluation with a pipeline installation task under four distinct scenarios. Given that the response time of ChatGPT is consistently swift, we employ the success rate across repeated scenarios as a reliable measure of system efficiency. We should note that the pipeline installation could also be considered as a case study for the flexibility test.

The remainder of this paper is structured as follows: Section 2 reviews the existing literature on construction robotics for assembly tasks and sequential planning for multi-step operations, and explores the potential applicability of large language models (LLMs) in addressing planning tasks. In Section 3, we introduce our novel system, RoboGPT, detailing its implementation for automating sequence planning in the realm of construction assembly. Section 4 presents the experimental results and evaluation of RoboGPT, which is enriched with insights from two case studies. In Section 5, we further evaluate RoboGPT's performance through a comparative study that involves designing pipeline connections in two distinct scenarios using two different sets of pipes. Finally, Section 6 provides a discussion on the broader implications of our findings, as well as suggestions for potential future research trajectories.

2. Related Work

2.1. Construction Robotics for Assembly Tasks

In recent years, the adoption of robotics in the construction industry has grown with the aim of improving efficiency, reducing labor costs, and enhancing safety on construction sites. This literature review explores various robotic systems and approaches that have been developed for construction assembly tasks, highlighting their advantages and challenges. The early research in construction robotics focused on developing specialized robotic systems for specific tasks. One example is the masonry robot SAM (Semi-Automated Mason) developed by Construction Robotics [13], which automates the bricklaying process, leading to reduced labor costs and increased productivity. Another example is the Ty Bot by Advanced Construction Robotics, a rebar-tying robot that streamlines the reinforcement process in concrete construction [14]. However, these specialized robotic systems often lack the flexibility to adapt to the dynamic and complex nature of construction environments. As a result, researchers have explored the use of more versatile robotic systems, such as modular robots and robotic arms, which can be reconfigured and programmed to perform various construction tasks. One notable example is a modular robotic system capable of autonomously assembling truss structures [15]. Similarly, Apolinarska, Pacher, Li, Cote, Pastrana, Gramazio, and Kohler [16] demonstrated the use of an industrial robotic arm in assembling complex timber structures, highlighting the potential for large-scale applications of robotic arms in construction. The emergence of digital fabrication techniques, such as 3D printing, has also influenced the development of construction robotics. A well-known example is the MX3D project, which utilized a robotic arm to 3D print a steel pedestrian bridge in Amsterdam [17]. Another study by Oke, Atofarati, and Bello [18] presented the Digital Construction Platform (DCP), a mobile robotic system capable of 3D printing building components on-site, offering a flexible and scalable approach to automated construction. The integration of robotic systems with Building Information Modeling (BIM) has been another area of interest for researchers. BIM, as a digital representation of a building's physical and functional characteristics, provides a wealth of information that can be utilized by robotic systems to plan and execute construction tasks. For instance, Gao, Meng, Shu, and Liu [19] proposed a BIM-based robotic assembly system for prefabricated building components, demonstrating the potential for enhanced efficiency and accuracy in construction processes. Collaborative robotics is another important area in construction assembly tasks, where multiple robots work together to achieve a common goal. Carey, Bardunias, Nagpal, and Werfel [20] demonstrated a swarm of construction robots inspired by termite behavior that can collaboratively build structures without centralized control. Similarly, Ding, Dwivedi, and Kovacevic [21] showcased a multi-robotic system for wire

arc additive manufacturing, highlighting the advantages of distributed robotic systems in construction.

Despite these advances in construction robotics, several challenges remain, including the need for robust perception and decision-making capabilities. To address these challenges, researchers have explored the integration of advanced computer vision and artificial intelligence (AI) techniques. For example, Zhang, Shen, and Li [22] developed a computer vision-based method for autonomous rebar picking and placing using a robotic arm, while Osa and Aizawa [23] demonstrated the use of deep learning algorithms for automating excavation tasks. Another primary challenge is the adaptability and versatility of robotic systems in varying construction environments. As stated by Ardiny, Witwicki, and Mondada [24], construction sites are dynamic and often unpredictable, making it difficult for robotic systems to perform tasks efficiently without constant human intervention. Another limitation is the high cost of developing and implementing advanced robotic systems, which may not be feasible for smaller construction firms. Furthermore, the integration of robotic systems requires extensive training for construction workers, which can be time consuming and costly [25]. Moreover, robotic systems often struggle with tasks that must be conducted with a proper sequence order of a series of steps, or assembly sequence planning (ASP) [26], which is the main focus of our investigation and will be discussed in depth in the next section.

2.2. Sequence Planning for Multi-Step Operations

Sequence planning for multi-step operations is largely addressed by the assembly sequence planning (ASP) literature, which is a critical aspect of manufacturing and construction that aims to identify the most efficient and cost-effective sequence of operations to assemble a product while considering various factors, such as resources, constraints, and goals [26]. Classical approaches to ASP include graph-based methods, such as the AND/OR graph [27] and the liaison graph [28], which represent assembly operations as directed graphs with nodes representing parts and edges. Matrix-based methods, such as the design structure matrix (DSM) [29] and the assembly incidence matrix (AIM) [30], use matrices to represent relationships between parts and assembly operations. Expert systems, like the blackboard system [31] and the CLIPS-based approach [32], employ human expert knowledge in the form of rules to generate assembly sequences. Additionally, mathematical methods like integer programming (IP) [33], mixed-integer linear programming (MILP) [8], and constraint programming (CP) [34] have been used to model and solve ASP problems by formulating them as mathematical models with variables representing assembly operations and constraints representing precedence relationships and resource limitations. Heuristic methods such as greedy algorithms [35]; local search methods like simulated annealing [36], tabu search [37], and variable neighborhood search [38]; constructive heuristics like the minimum degree heuristic [39]; and decomposition methods [40] have been employed to find good solutions efficiently by using simple rules and shortcuts to navigate the complex solution space of ASP problems. These heuristic methods are typically faster and less computationally demanding compared to optimization techniques, making them suitable for large-scale assembly problems. In recent years, modern approaches leveraging computational power and artificial intelligence have emerged, such as genetic algorithms [41], ant colony optimization [42], particle swarm optimization [43], and artificial neural networks [9]. These methods can be used to explore a wide search space and generate optimal or near-optimal solutions.

However, these methods have certain limitations that hinder their effectiveness in addressing the dynamic nature of construction projects. Most ASP methods assume deterministic input data and fail to handle uncertainties related to structure dimensions, assembly resources, and process variability. In real-world production environments (e.g., manufacturing and construction), various sources of uncertainty may arise, such as geometric and material property variations, tool capabilities, resource availability, and human operator performance [44]. As such, to facilitate construction tasks that utilize sequence

planning, mathematical and heuristic techniques will need to rely on algorithms tailored by human experts that leverage domain-specific knowledge and rules. While these methods can effectively navigate the complex solution space for complex and variable construction tasks, they may impose a significant computational overhead due to the need for continuous adaptation and refinement of the heuristics as the construction process evolves [45]. This limitation may lead to suboptimal solutions in highly dynamic and uncertain environments, where rapid adjustments to changing circumstances are essential [46]. On the other hand, although machine learning methods, such as genetic algorithms and neural networks, can adapt to dynamic scenarios much easier compared to mathematical and heuristic techniques, they require a significant amount of training data to achieve accurate results [11]. In construction operations, where site conditions and project requirements can change frequently, acquiring sufficient training data for every possible scenario is challenging, which limits the adaptability of these methods to dynamic environments [47]. Moreover, the performance of machine learning methods may be negatively affected by the presence of noisy or incomplete data [48], which is often the case in construction projects.

2.3. LLMs for Sequence Planning

Large language models (LLMs), such as OpenAI's GPT series [49], BERT [50], and T5 [51], have demonstrated impressive performance in natural language understanding, generation, and reasoning tasks. These models, based on the Transformer architecture [52], utilize deep learning techniques and massive datasets to learn contextual representations of text, allowing them to generate coherent and contextually relevant responses. The application of large language models in sequence planning tasks, such as ASP, remains relatively unexplored. However, recent studies have started investigating the potential of these models for various sequence planning tasks. For example, LLMs have been used to generate textual descriptions of process plans based on given input constraints [53]. These models can potentially be employed to generate high-level assembly plans or provide guidance to human operators during the assembly process. In domains like project management and logistics, large language models have been used to generate natural language descriptions of optimal schedules or resource allocations [54]. Large language models have demonstrated their potential in generating task plans for robotic systems based on natural language instructions [55,56]. These models can be adapted to generate action sequences for robotic assembly operations or other complex robotic tasks, providing a more intuitive human-robot interaction experience. This application can be extended to sequence planning tasks, such as generating assembly schedules or allocating resources for assembly operations. In a recent study by Prieto, Mengiste, and García de Soto [54], ChatGPT-3.5 was used to generate a construction schedule for a simple construction project. The output from ChatGPT was evaluated by a pool of participants who provided feedback on their overall interaction experience and the quality of the output. The results showed that ChatGPT could generate a coherent schedule that followed a logical approach to fulfill the requirements of the scope indicated. The participants had an overall positive interaction experience and indicated the potential of such a tool in automating many preliminary and time-consuming tasks. Vemprala, Bonatti, Bucker, and Kapoor [57] conducted an experimental study exploring the use of OpenAI's ChatGPT for robotic applications. The study proposed a strategy that combines design principles for prompt engineering and the creation of a high-level function library, enabling ChatGPT to adapt to different robotic tasks, simulators, and form factors. The evaluations focused on the effectiveness of various prompt engineering techniques and dialogue strategies in executing diverse robotic tasks.

As for the reasons why ChatGPT can be used for sequence planning, these are still largely unknown. Earlier evidence shows that ChatGPT has inherent capabilities as a large-scale language model, as it has been trained on massive amounts of text data, enabling it to understand and generate human-like text [12]. This capability may allow it to comprehend the requirements of a sequence planning problem and generate potential solutions in a human-readable format. In addition, it seems that ChatGPT can capture context and reason

about the relationships between different elements in a sequence planning problem. It can consider constraints, dependencies, and objectives to generate effective solutions [58]. ChatGPT’s extensive training data allows it to possess a broad range of knowledge across various domains. This knowledge could be potentially leveraged to address sequence planning tasks in multiple industries, such as manufacturing, construction, logistics, and robotics [59]. The exploration of this study in applying ChatGPT in complex sequence planning for robotics is expected to add more empirical evidence and lead to new methods for automation.

3. Methodology

3.1. Architecture

Figure 1 presents the comprehensive system architecture of RoboGPT, which is composed of four primary components: the Robot Control System, Scene Semantic System, Object Matching System, and User Command Decoder System. ChatGPT, an advanced natural language processing model, functions as the central intelligence within the system. Upon receiving task descriptions and specific requirements from users, ChatGPT meticulously generates sequential solution commands in a step-by-step manner, adhering to the precise requirements of the task. The generated response text is subsequently decoded by the User Command Decoder System and transmitted to a Unity-based virtual environment in the form of virtual objects. The Scene Semantic System is responsible for detecting real-world objects, which are then sent to the Unity environment to be meticulously aligned and matched with their virtual counterparts. Once the alignment is complete, the objects, in conjunction with the corresponding actions derived from the commands, are relayed to the Robot Control System to facilitate real-world object manipulation.

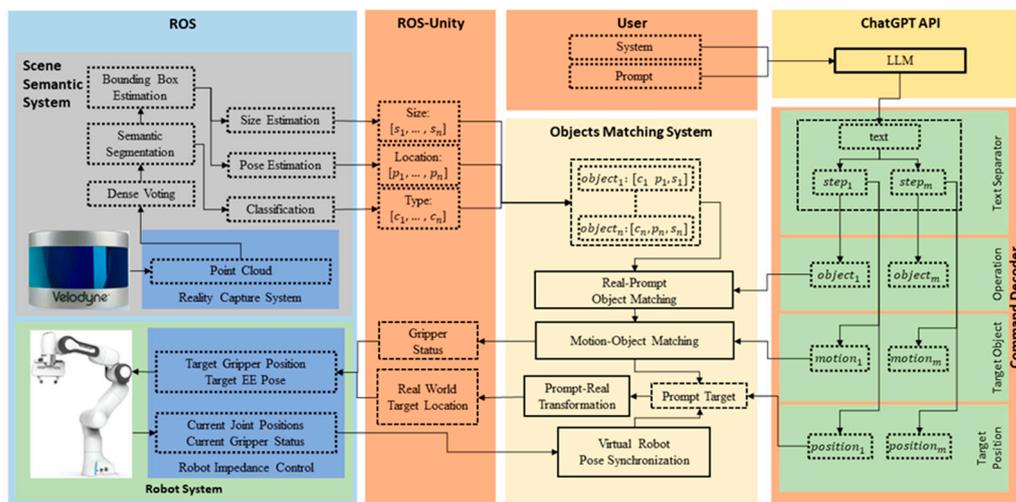


Figure 1. System structure.

3.2. Robot Control System

The robot system tested in this study was a Franka Emika Panda robot arm, which is a lightweight, compact, and versatile robot designed for human–robot collaboration and which is widely used in manufacturing, research, and education as it is known for its ease of use, flexibility, and reliability. The Panda has seven degrees of freedom corresponding to its seven joints. Each joint is equipped with a force/torque sensor and a joint-angle sensor to accurately measure the states of the robot arm, allowing it to move in various directions and perform intricate tasks with high precision. A parallel gripper is attached as the end-effector on the seventh joint, which can be used to interact with objects through picking up and dropping.

To smoothly control the end-effector and generate a stable moving trajectory, the impedance controller is applied in cartesian coordinates, as shown in Figure 2. The

impedance of the end-effector can be adjusted based on the force or torque applied by the environment, allowing the robot to adapt to varying conditions. Specifically, the controller imposes spring–mass–damper behavior on the mechanism by maintaining a dynamic relationship between the force, position, velocity, and acceleration:

$$F = C \cdot v_{ee} + K \cdot \Delta x_{ee}^{rob} + l \cdot d, \quad (1)$$

where F , v_{ee} , Δx_{ee}^{rob} , l , and $d \in \mathbb{R}^3$ are the implemented force on the end-effector, the velocity of the end-effector, and the position of the end-effector in the robot coordinate system and payload, respectively. Given the end-effector's current position $x_{ee_cur}^{rob}$ and the desired position $x_{ee_desire}^{rob}$, Δx_{ee}^{rob} is calculated as:

$$\Delta x_{ee}^{rob} = x_{ee_desire}^{rob} - x_{ee_cur}^{rob}, \quad (2)$$

$x_{ee_desire}^{rob}$ is the real-world target location derived from the Real-Virtual Object Matching System that is discussed in the following section. In order to control the virtual robot arm in Unity to interact with the virtual objects, the real-time joint position $\mathbf{q}^{rob} \in \mathbb{R}^3$ is sent to Unity through the ROS-Unity bridge (RUB) to synchronize the virtual arm. Each element in \mathbf{q}^{rob} is the rotation angle for the corresponding joint. The gripper's status of the real robot arm is also sent through the RUB to instruct the virtual robot's behavior, and the interaction between the virtual gripper and objects will be sent back to ROS to control the real gripper's action.

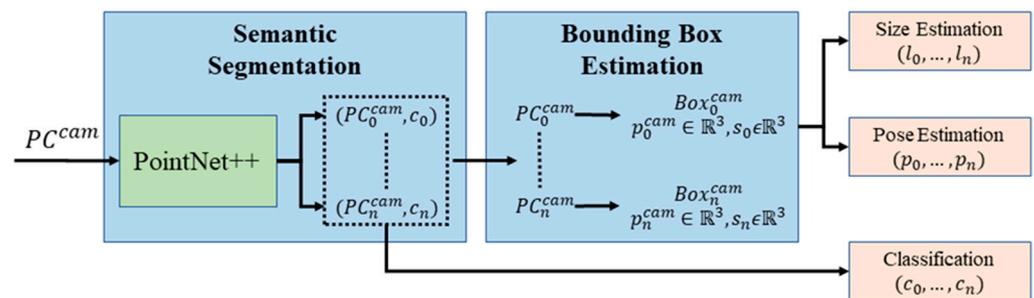


Figure 2. Scene Semantic System.

3.3. Semantic Segmentation System

The Scene Semantic System collects the visual information from the surrounding environment and detects the real target objects for downstream alignment. A Velodyne-16 LiDAR (VL16) is used to capture point cloud data and save it on the ROS platform. The LiDAR sensor coordinate system is calibrated with the Panda coordinate system to ensure the positions of the detected objects.

The VL16 was selected as the scanning sensor because of its high scanning speed and stable scanning results. Since the VL16 only has 16 scanning rings in the vertical direction, which is too sparse to capture the detail spatial information, an augmentation scanning strategy was applied to register the scanning results from multiple viewpoints and generate a dense scanning result. To eliminate the influence of the error caused by the registration of multiple frames, we applied the density-voting clustering method to shift the drifting points to the closest density center so that all the returning points will be close to the object surfaces and the shape of the objects can be perfectly captured.

The virtual scene data, including joint states, point clouds, and virtual objects with physical properties, are then sent to the Unity game engine for interface reconstruction. In order to subscribe to data from ROS via the network, the ROS-Unity bridge and ROS# are used to build a WebSocket, which allows two-way communication and data transfer between ROS and Unity. We also used ROS# to build some nodes in Unity to publish and subscribe to topics from ROS. Baxter's state data (URDF, joint, and gripper state) are used to build a virtual Baxter that replicates the same states of the real Baxter. The same

prefab library as mentioned in the scene recognition system is used to provide virtual object information with physical properties that can be used to rebuild stationary objects in the game engine. We also use the Unity physical engine to assign the point cloud and virtual object with physical properties and rebuild a virtual working scene based on the data from ROS.

The augmented and clustered point cloud $PC^{cam} \in R^{N \times 3}$ is then fed into PointNet++, which we took as our segmentation model as shown in Figure 2. N denotes the number of points according to the input size of the model. PointNet++ is a deep learning model that has been well-trained on various point cloud datasets and can handle both object detection and semantic segmentation tasks. In this application, we only focused on the segmentation branch of PointNet++ to obtain the object labels of each point. The segmented points are clustered as point sets $[PC^{cam}_0, \dots, PC^{cam}_n]$ and the corresponding predicted labels $[c_0, \dots, c_n]$. The point sets are then used to estimate the oriented bounding boxes that closely wrap all of the points as $[Box^{cam}_0, \dots, Box^{cam}_n]$. The bounding boxes are parameterized as $Box^{cam}_i := [s_i^T, p_i^T]^T$, where s_i and $p_i \in R^3$ are the size (width, length, and height) and location (x_i , y_i , and z_i). The labels, sizes, and locations of segmented point sets are then sent to Unity through the RUB as the classification results, size estimation results, and pose estimation results.

3.4. Command Decoder

The command decoder works as the translator to transfer the response from ChatGPT in natural language into a machine-understandable programming command so that the robot arm can execute the actual sequential actions inferred by ChatGPT. We used the ChatGPT-4 model and coded with python and C# to build the API to communicate between Unity and the online model. The API is based on the HTTP request. The user sends a text prompt to the API, and it will return a response in the form of a text message. The API also supports various customization options to regularize the response by typing the specific requirements in the “system” section.

For most construction assembly tasks, the sequential actions can be simplified as moving an object to a certain location. For example, moving the pipe to position A or putting the brick at position B. Therefore, the operation command for the sequential actions can be represented by an action, object, and target position. In order to make the reply from ChatGPT more explainable, we designed the “system” with three principles:

1. ChatGPT will generate the reply step by step in an execution order.
2. For each step, there is only one motion and one object to be moved or operated. There is only one target location.
3. The related words about action, object, and target position must be surrounded by brackets.

Given the regularization principles, the reply from ChatGPT could be simplified as:

1. [Action 1] [object 1] to [position 1].
2. [Action 2] [object 2] to [position 2].
- ...

Therefore, the regularized reply from ChatGPT can be firstly split into single steps. Then, the single steps can be used to extract the action, object, and target position, as shown in Figure 3. The brackets are used to crop the action or object names as strings. The detected string will then be checked to see if it shows up in the pre-defined action or object dictionary. If the dictionaries contain the string, the corresponding action or object will be sent to the Real-Virtual Object Matching System. The dictionary contains the name of common actions and objects in construction sites.

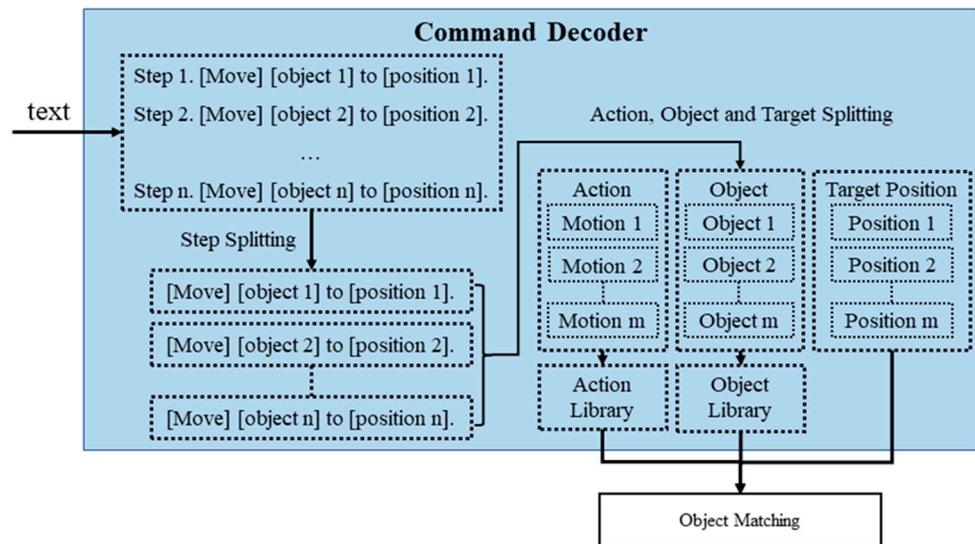


Figure 3. Command Decoder System structure.

3.5. Object Matching System

The detected action, object, and position will then be sent to the Object Matching System to be paired with the detected objects from the real world and to be transferred to the robot arm control codes. Specifically, the detected object, noted as $\text{obj}_i^{\text{prompt}}$, will be firstly matched with the label of segmented objects from the Semantic Segmentation System, noted as $\text{obj}_j^{\text{seg}}$. Note that the labels of the segmentation system are strings that are included in the object dictionary. Given a matched pair $(\text{obj}_i^{\text{prompt}}, \text{obj}_j^{\text{prompt}})$, $\text{obj}_i^{\text{prompt}}$ is then assigned the parameters of $\text{obj}_j^{\text{seg}}$, including l_j^{seg} for size and p_j^{seg} for position. Then, $\text{obj}_i^{\text{prompt}}$ has four major properties:

$$\text{obj}_i^{\text{prompt}} := [l_i^{\text{prompt}}, p_i^{\text{prompt}}, \text{action}_j^{\text{prompt}}, \text{position}_i^{\text{prompt}}], \quad (3)$$

where $\text{action}_j^{\text{prompt}}$ is the matched action from the dictionary and $\text{position}_i^{\text{prompt}}$ is the target location. Thus, the desired operation on $\text{obj}_i^{\text{prompt}}$ is parameterized as its current position, the action, and its target position, which can be understood by the robot arm. Then, $\text{obj}_i^{\text{prompt}}$ can be sent to the Panda for a single-step operation in a sequence.

4. Test Cases

In this section, we report three cases to showcase how the proposed RoboGPT system accomplishes assembly tasks, including material stacking, a Hanoi tower puzzle, and plumbing assembly. The detailed results of the test cases are detailed in the following sub-sections.

4.1. Material Stacking

The first case, object stacking, is related to placing materials based on their sizes. A real-world example is the stacking of construction materials on a site based on their dimensions and shapes for a stable structure. We used four disks of different sizes as an example. The proposed system was requested to stack the disks as stable as possible. The prompt we gave to the ChatGPT was: “I have five cubes with names [A] to [E]. The cubes’ lengths are in a descending order from [A] to [E]. So, I want to teach a robot arm to use the cubes to create a tower with the most stable design. Could you tell me which cube to operate step by step?”

ChatGPT analyzed the size and weight of the disks and generated an assembly sequence, starting with the largest disk at the bottom and placing the others in decreasing order of size as follows:

1. Move cube [A] to the base location where the tower will be built;
2. Move cube [B] from its original location to the top of cube [A];

3. Move cube [C] from its original location to the top of cube [B];
4. Move cube [D] from its original location to the top of cube [C];
5. Move cube [E] from its original location to the top of cube [D].

The UI in Unity is shown in Figure 4. Clicking the button on the top right will send the prompt from Unity to the Python ChatGPT-4 API, as shown in Figure 5. Then, the deployed ChatGPT-4 API sends the generated assembly sequence to the robot arm (Franka Emika Panda) for locomotion controls.

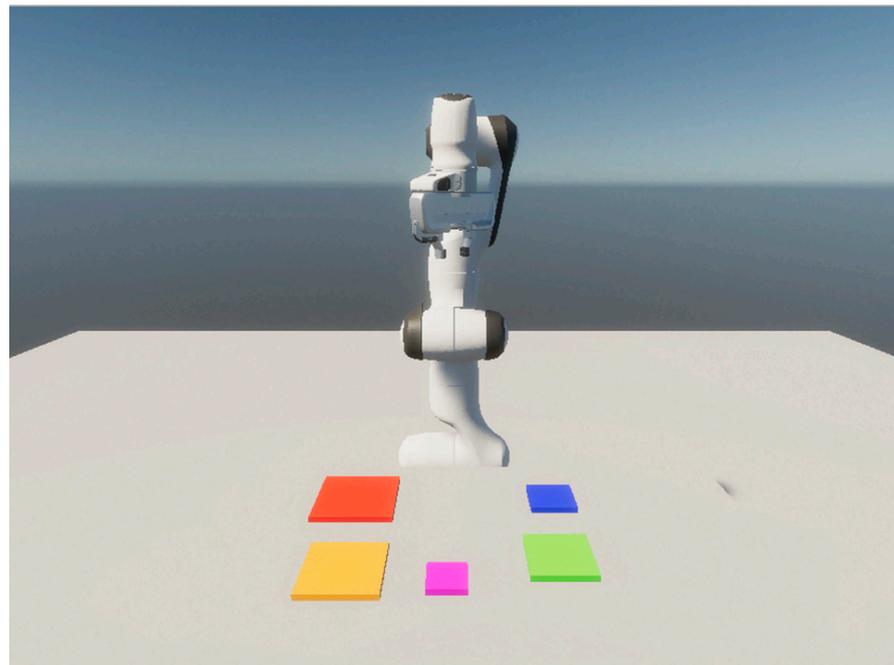


Figure 4. ChatGPT-4 user interface in Unity for material stacking.

```
(chatgpt_unity) D:\Hengxu\Python-Unity-Socket-Communication>python chatgpt_unity.py
User: I have five cubes with the names [A] to [E]. The cubes' lengths are in a descending order from [A] to [E]. So I want to teach a robot arm to use the cubes to create a tower with the most stable design. Could you tell me which cube to operate step by step and use "move" to refer the operations?

(chatgpt_unity) D:\Hengxu\Python-Unity-Socket-Communication>python chatgpt_unity.py
User: I have five cubes with the names [A] to [E]. The cubes' lengths are in a descending order from [A] to [E]. So I want to teach a robot arm to use the cubes to create a tower with the most stable design. Could you tell me which cube to operate step by step and use "move" to refer the operations?
ChatGPT: Sure, here are the steps to build the tower using the robot arm:

1. Move cube [A] to the base location where the tower will be built.
2. Move cube [B] from its original location to the top of cube [A].
3. Move cube [C] from its original location to the top of cube [B].
4. Move cube [D] from its original location to the top of cube [C].
5. Move cube [E] from its original location to the top of cube [D].

Note that each move needs to be carefully planned to ensure the stability of the tower. The robot arm should move slowly and precisely to avoid knocking over any of the cubes.
```

Figure 5. ChatGPT-4 response for robot control in material stacking.

The computer vision module, utilizing a combination of object detection and recognition algorithms, was used to identify the five disks and their positions in the workspace. We simply used five obvious labels to help the robot arm to locate the objects since we did not focus on building object detection algorithms. With this information, the system calculated the necessary movements for the robot arm to execute the stacking task based on the described sequence order. By precisely controlling the position, orientation, and gripping force of the robot arm, the system successfully stacked the five cubes in a stable arrangement, demonstrating the effectiveness of ChatGPT-4 in guiding robotic systems to accomplish complex assembly tasks. Figure 6 shows the simulated robot arm placing the objects in Unity. Figure 7 shows the Franka Emika Panda robot arm placing the blocks to

form a tower in the real world. For each step, the sub-figure on the left shows the real robot arm's action and the sub-figure on the right shows the pose of the robot arm in ROS.

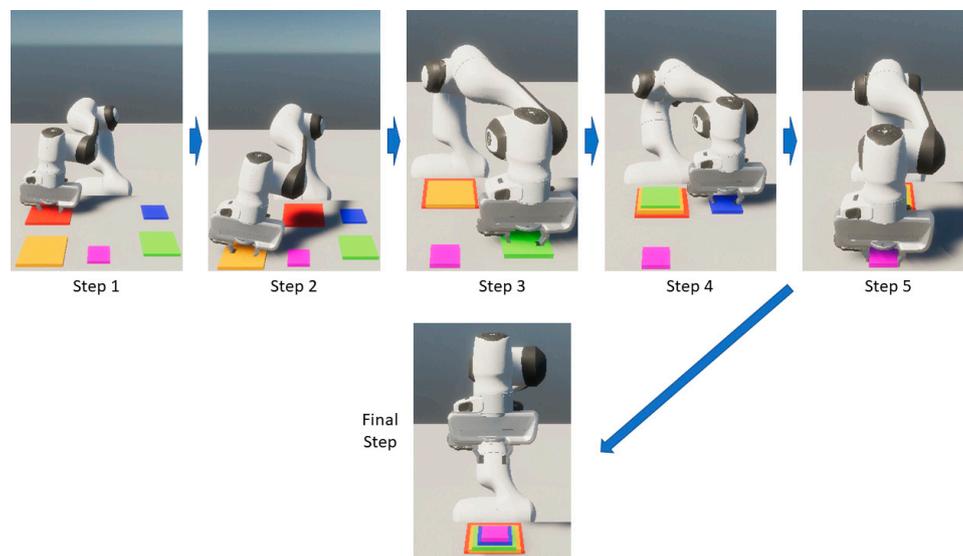


Figure 6. Simulated robot arm building the 5-cube tower.

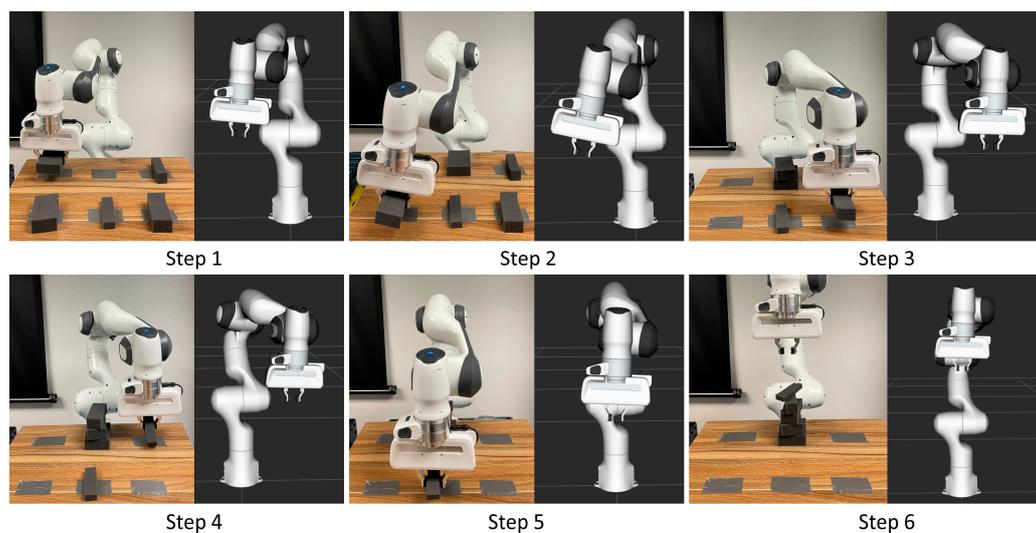


Figure 7. Real robot arm stacking 5 soft blocks.

4.2. Hanoi Tower Puzzle

The second case we tested was the classic tower of Hanoi puzzle, using four disks of different sizes. ChatGPT was used to generate an optimal solution, which involved moving the disks among three pegs while adhering to the puzzle's rules: only one disk can be moved at a time, and a disk cannot be placed on top of a smaller disk. The prompt we provided was: *"I have a tower of Hanoi with five disks [A], [B], [C], [D], [E], from smallest to biggest. Describe the sequence of completing the puzzle and control the robot arm to finish it."*

ChatGPT formulated a step-by-step sequence to solve the puzzle, guiding the robot to move the disks between the pegs in the correct order. The robot successfully completed the task, demonstrating the ability of ChatGPT to solve complex problems with specific constraints. In this case, the ChatGPT-4 API was deployed to solve the Hanoi tower puzzle using the robot arm (Franka Emika Panda). Figure 8 shows the same UI in Unity and the setup for the Hanoi tower puzzle problem. Figure 9 shows the prompt sent from Unity to Python and the sequence reply from ChatGPT. By accurately controlling the position,

orientation, and gripping force of the robot arm, the system was able to successfully stack the four disks in a stable arrangement.

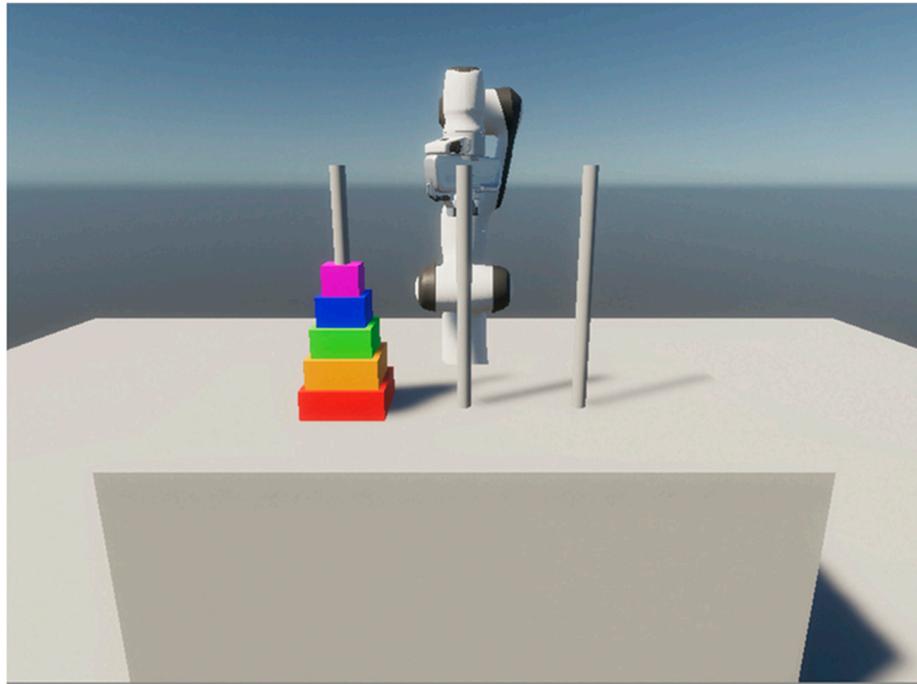


Figure 8. ChatGPT-4 user interface in Unity for Hanoi tower puzzle.

```
(chatgpt_unity) D:\Hengxu\Python-Unity-Socket-Communication>python chatgpt_unity.py
User: I have a tower of Hanoi with five disks [A], [B], [C], [D] and [E] from smallest to biggest. Describe the sequence
of completing the puzzle and control the robot arm to finish it.

(chatgpt_unity) D:\Hengxu\Python-Unity-Socket-Communication>python chatgpt_unity.py
User: I have a tower of Hanoi with five disks [A], [B], [C], [D] and [E] from smallest to biggest. The towers names are
[a], [b] and [c]. Describe the sequence of completing the puzzle and control the robot arm to finish it.
ChatGPT: 1. Move [A] from [a] to [b].
2. Move [B] from [a] to [c].
3. Move [A] from [b] to [c].
4. Move [C] from [a] to [b].
5. Move [A] from [c] to [a].
6. Move [B] from [c] to [b].
7. Move [A] from [a] to [b].
8. Move [D] from [a] to [c].
9. Move [A] from [b] to [c].
10. Move [B] from [b] to [a].
11. Move [A] from [c] to [a].
12. Move [C] from [b] to [c].
13. Move [A] from [a] to [b].
14. Move [B] from [a] to [c].
15. Move [A] from [b] to [c].
16. Move [E] from [a] to [b].
17. Move [A] from [c] to [a].
18. Move [B] from [c] to [b].
19. Move [A] from [a] to [b].
20. Move [C] from [c] to [a].
21. Move [A] from [b] to [c].
22. Move [B] from [b] to [a].
23. Move [A] from [c] to [a].
24. Move [D] from [c] to [b].
25. Move [A] from [a] to [b].
26. Move [B] from [a] to [c].
27. Move [A] from [b] to [c].
28. Move [C] from [a] to [b].
29. Move [A] from [c] to [a].
30. Move [B] from [c] to [b].
31. Move [A] from [a] to [b].
```

Figure 9. ChatGPT-4 response for robot control in Hanoi tower puzzle.

The generated assembly sequence was sent to the robot arm for locomotion controls. Similarly, a computer vision module, consisting of object detection and recognition algorithms, was employed to identify the five disks and the three towers along with their positions within the workspace. To simplify the system design, we used three square labels to represent the towers and only required the real robot arm to recognize the location of the towers. Upon obtaining the positions of the disks, the system calculated the necessary robot arm movements to execute the stacking task according to the sequence order provided by ChatGPT-4. Figure 10 shows the step-by-step motion of the simulated robot arm in

Unity when solving the 5-disk Hanoi tower puzzle. The response from ChatGPT-4 was the optimal answer, consisting of a 31-step sequence, and the robot arm could precisely follow the instructions from ChatGPT and complete the problem. This test further proved that the proposed RoboGPT system can effectively solve a sequential learning problem and interact with real objects to solve the problem in the real world.



Figure 10. Simulated robot arm solving the Hanoi tower puzzle.

5. Comparison Study

In order to demonstrate the advantages of the proposed RoboGPT system in intricate multi-stage robotic operations and investigate the capacity of ChatGPT to address real-world construction challenges, we conducted a comprehensive evaluation of the RoboGPT system in the context of a pipeline installation under various conditions. This comparative study aimed to assess the system's performance, as well as to elucidate its potential and limitations.

We opted not to incorporate the material stacking and Hanoi tower puzzle scenarios in this investigation for two primary reasons. Firstly, the material stacking task is relatively elementary, as it predominantly necessitates rudimentary knowledge of object stacking based on size. Secondly, the central challenge of the Hanoi tower puzzle resides in completing the task within a constrained timeframe, which does not align with the objectives of our study.

Conversely, the pipeline installation scenario presented a more open-ended challenge, requiring the system to determine the spatial dimensions, evaluate resource availability, and devise an appropriate method for connecting the pipes. It is crucial to note that this task does not entail a singular solution; rather, multiple viable solutions can achieve the desired outcome. Consequently, the pipeline installation task, which demands a thorough assessment of dimensions, resource estimation, and sequencing while considering both spatial and resource constraints, is better suited for our comparative analysis. We applied two different tasks with two different conditions to evaluate the performance of the purposed system. Since pipe installation tasks in the real world often require a large spatial space, which is hard to manipulate with a research-based robot arm, we built the simulation environment in Unity to test the results. Given the knowledge from a pipe installation process at a real construction site, we designed the Avoid Obstacles and Pass Points tasks for further testing.

The Avoid Obstacles task is to design a pipeline that connects two points, but the pipes cannot pass certain points. This task was designed to simulate a case where the pipes have to avoid some pre-built structures or safety areas. The testing environment was designed as a $10 \times 10 \times 10$ room with a start point location of $P_{\text{start}}^1 = (5, 5, 0)$ on the floor and the end point of $P_{\text{end}}^1 = (5, 5, 10)$ on the roof. The two obstacle points, named A_{Obs} and B_{Obs} , were located at $(5, 5, 5)$ and $(5, 7, 5)$, respectively. Figure 11 shows the setup environment of Avoid Obstacles. The green cube denotes the start point and the red cube denotes the end point. The two small black cubes denote the obstacles to be avoided.

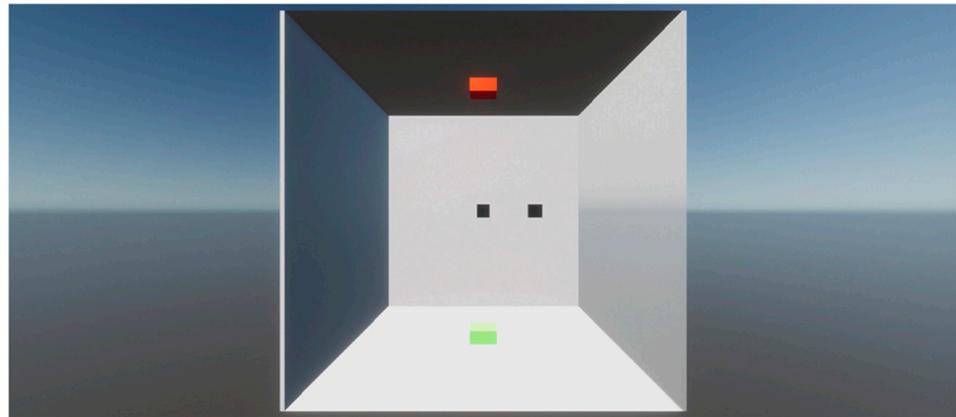


Figure 11. The setup environment of Avoid Obstacles task.

The Pass Points task is to find a solution and design a pipeline between two given positions, passing certain points. This situation was designed to simulate a case where the pipes must connect some devices, such as air conditioners, or the pipes must pass through some holders on the wall as supports. Similarly, the testing environment was also in a $10 \times 10 \times 10$ room with the start point location of $P_{\text{start}}^2 = (0, 0, 0)$ and the end point of $P_{\text{end}}^2 = (10, 10, 10)$. The two mandatory points, named A_{man} and B_{man} , were located at $(0, 0, 8)$ and $(6, 6, 0)$, respectively. Figure 12 shows the setup environment of Pass Points. Similarly, the green cube denotes the start point and the red cube denotes the end point. The two small black cubes denote the mandatory points to be connected.

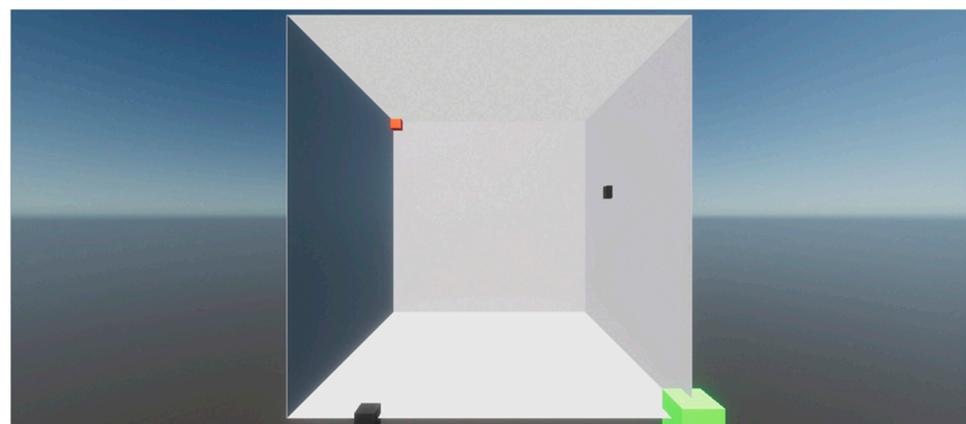


Figure 12. The setup environment of Pass Points task.

5.1. Avoid Obstacles Task

In the Avoid Obstacles task, we set two different conditions: the constant condition and variable condition. To be specific, the constant condition refers to the situation where the pipes to be used to build the pipeline are the same size. In our case, we set the length of the pipes to be 2. Note that the diameter of the pipes was ignored. On the contrary, the

variable condition referred to the case where the pipes' sizes were not fixed. To be specific, we used three types of pipes with the lengths of 2, 3, and 4. The system can choose any of the pipes to build the pipeline.

The prompt we used for the constant condition is as follows: "Can you help me with pipe connection? We have several 2ft length straight pipes (pipe 2ft), 3ft length straight pipes (pipe 3ft), 4ft length straight pipes (pipe 4ft). The start position is (5ft, 5ft, 0ft) direction is the positive Z axis, the end position (5, 5, 10) direction is the negative Z axis. We assume that each straight pipe can be connect to each other directly. You can just tell me the position of each pipe, such as 'pipe 2ft #1 (5, 5, 2) z axis, pipe 2ft #2 (5, 5, 4) z axis, pipe 2ft #3 (5, 7, 4) y axis'. To be noted, each pipe must maintain parallelism to the X, Y, and Z axes. There are two obstacles at point (5, 5, 5) and point (5, 7, 5), the pipe cannot pass through this point from neither X, Y nor Z axes."

The prompt for the variable condition is: "Can you help me with pipe connection? We have several 2ft length straight pipes (pipe 2ft), 3ft length straight pipes (pipe 3ft), 4ft length straight pipes (pipe 4ft). The start position is (0ft, 0ft, 0ft) direction is the positive Z axis, the pipe connection must pass the first mandatory point (0, 0, 8), then pass the second mandatory point (6, 6, 0), finally to the end position (10, 10, 10) direction is the negative Z axis. We assume that each straight pipe can be connect to each other directly. You can just tell me the position of each pipe, such as 'pipe 2ft #1 (0, 0, 2) z axis, pipe 4ft #1 (0, 0, 6) z axis, pipe 3ft #1 (0, 3, 6) y axis'. To be noted, each pipe must maintain parallelism to the X, Y, and Z axes. The pipe must pass each mandatory point (0, 0, 8) and (6, 6, 0)."

For each condition, we used the same prompt to generate 20 trials. Table 1 lists the counting results of successful and failed trials. The sub-optimal trials refer to the cases where the RoboGPT system could give the correct connection design, but with unnecessary pipes and detours.

Table 1. Counting results of successful and failed trials.

Condition	Success		Fail	Total	Optimal /Total	Success /Total	Failed /Total
	Optimal	Sub-Optimal					
Constant	12	8	0	20	0.6	1	0
Variable	3	5	12	20	0.15	0.25	0.8

The results show a significant difference between the successful rates of the two conditions, being 100% for the constant condition and 25% for the variable condition. Theoretically, the two conditions corresponded to two difficulty levels in solving the problem. For the first condition, the pipe's length is almost the unit length compared with the room's scale. There is no need to consider the arrangement of pipes to achieve a certain length of the total pipeline. In other words, the final solution could use any number of pipes and the only requirement was to avoid A_{obs} and B_{obs} , and to finally reach P_{end} ¹. However, for the second condition, the lengths of the pipes vary from 2 to 4. So, the solution had to not only satisfy the requirement of passing the mandatory points and reaching the target, but also had to find the proper combination of pipes with different sizes to achieve the full length of the pipeline. This was an extra constraint which restricted the solution space, added logical difficulty, and made the problem harder to solve. In other word, the resource pipes that could be used to build the pipeline were restricted. Figure 13 shows the assembling process of a successful trial.

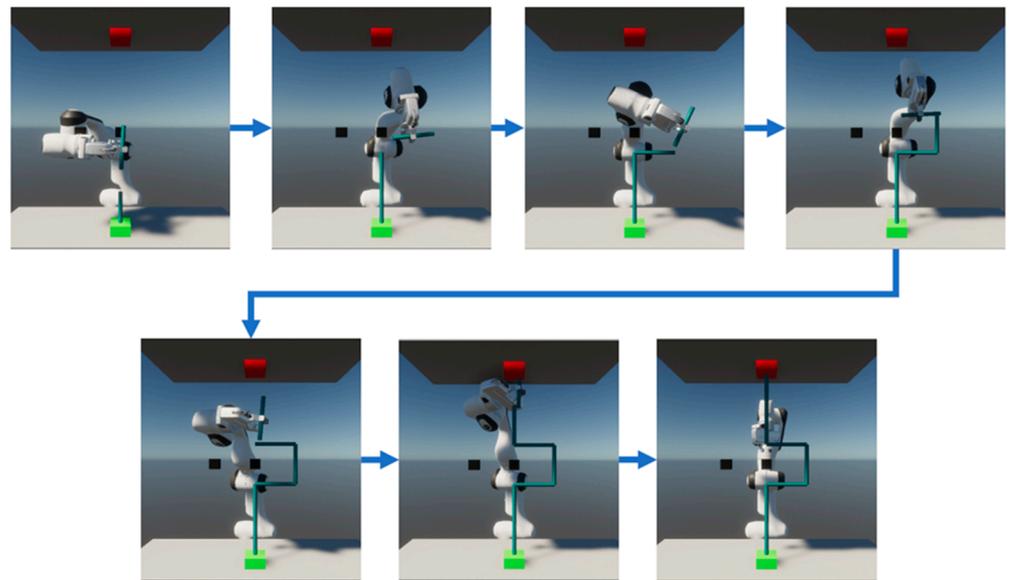


Figure 13. The assembling process of a successful trial for Avoid Obstacles task.

Figure 14 shows a typical sub-optimal solution demonstrating that, given sufficient pipes without any constraints, ChatGPT can give a redundant design with unnecessary costs. The proposed pipeline by ChatGPT makes an unnecessary detour to avoid the obstacles. Figure 15 gives a successful example as a standard solution of variable condition. Figure 16 illustrates the shortcomings of ChatGPT in spatial understanding. The layout on the left shows the failure in condition 2 as the pipe only reaches the height of the end point, but it does not find the location on the x-z plane. The failed layout on the right shows that the start and end point of the pipe was not understood, so the following pipe was connected from the middle of the previous pipe, as shown in the red circle. The results proved that ChatGPT cannot always precisely understand the spatial information from a pure text input.

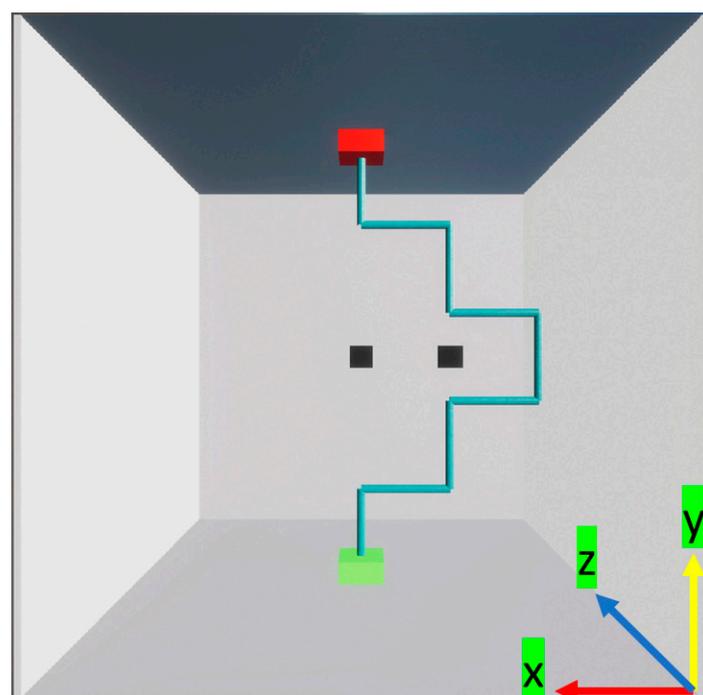


Figure 14. The layout of sub-optimal trial under constant condition.

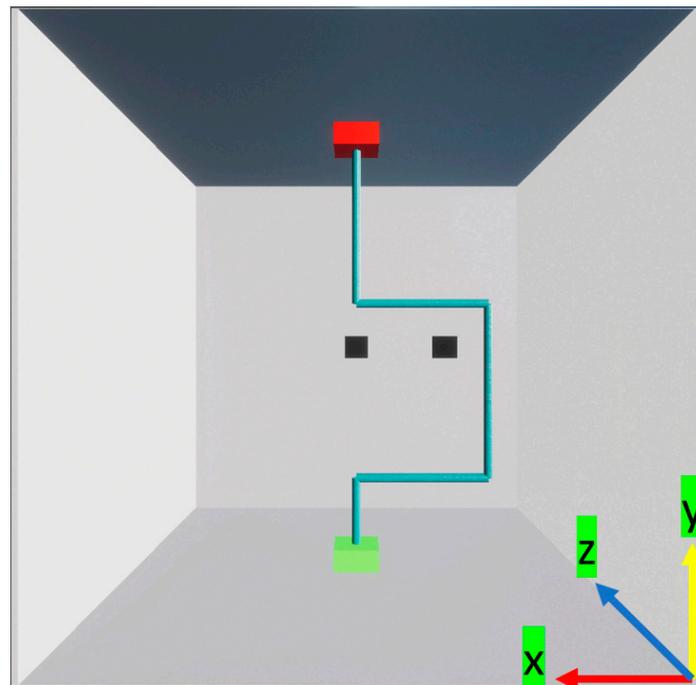


Figure 15. The layout of successful trial under variable condition for avoid obstacles task.

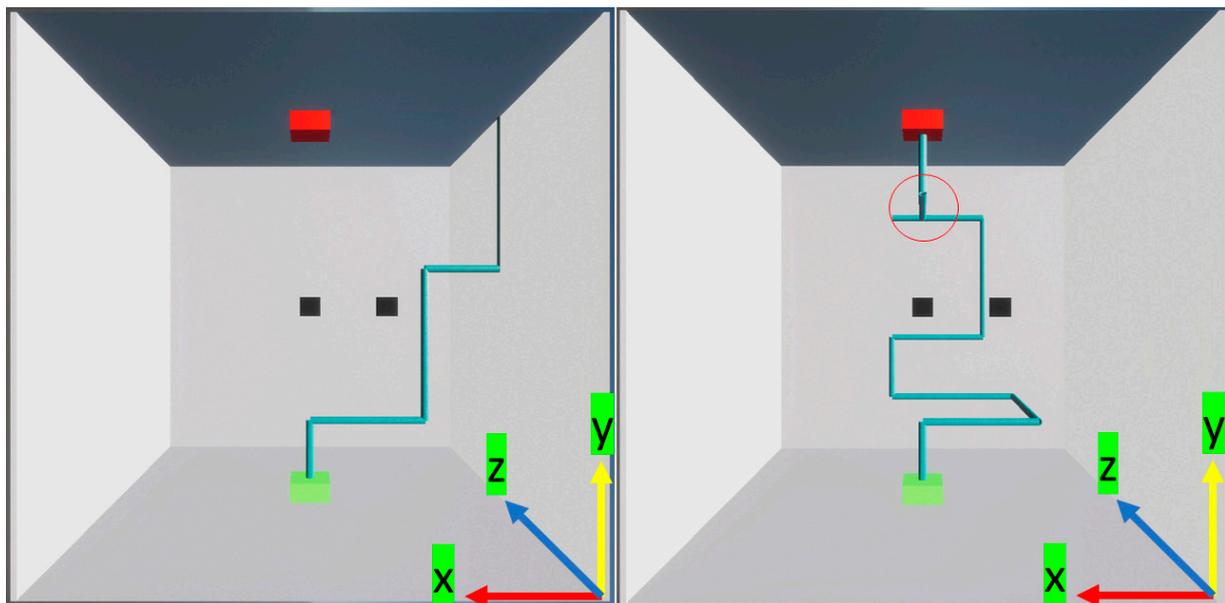


Figure 16. The layout of failed trial under variable condition for avoid obstacles task.

5.2. Pass Points Task

In the Pass Points task, we used the same two conditions as in the previous task. The prompt we used for the constant condition is as follows: “Can you help me with pipe connection? We have several 2ft length straight pipes (pipe 2ft). The start position is (0ft, 0ft, 0ft) direction is the positive Z axis, the pipe connection must pass the first mandatory point (0, 0, 8), then pass the second mandatory point (6, 6, 0), finally to the end position (10, 10, 10) direction is the negative Z axis. We assume that each straight pipe can be connect to each other directly. You can just tell me the position of each pipe, such as ‘pipe 2ft #1 (0, 0, 2) z axis, pipe 2ft #2 (0, 0, 4) z axis, pipe 2ft #3 (0, 2, 4) y axis’. To be noted, each pipe must maintain parallelism to the X, Y, and Z axes. The pipe must pass each mandatory point (0, 0, 8) and (6, 6, 0).”

The prompt for the variable condition is: “Can you help me with pipe connection? We have several 2ft length straight pipes (pipe 2ft), 3ft length straight pipes (pipe 3ft), 4ft length straight pipes (pipe 4ft). The start position is (0ft, 0ft, 0ft) direction is the positive Z axis, the pipe connection must pass the first mandatory point (0, 0, 8), then pass the second mandatory point (6, 6, 0), finally to the end position (10, 10, 10) direction is the negative Z axis. We assume that each straight pipe can be connect to each other directly. You can just tell me the position of each pipe, such as ‘pipe 2ft #1 (0, 0, 2) z axis, pipe 4ft #1 (0, 0, 6) z axis, pipe 3ft #1 (0, 3, 6) y axis’. To be noted, each pipe must maintain parallelism to the X, Y, and Z axes. The pipe must pass each mandatory point (0, 0, 8) and (6, 6, 0).”

Similarly, we used the same prompt to generate 20 trials with new chat channels. Table 2 lists the counting results of successful and failed trials. To intuitively show the results from the two conditions, we picked a success trial and a failed trail from each condition and provide the visualizations in Figures 17–20.

Table 2. The counting results of experiments on Pass Points task under two conditions.

Condition	Success	Fail	Total	Success /Total	Failed /Total
Constant	14	6	20	0.7	0.3
Variable	4	16	20	0.2	0.8

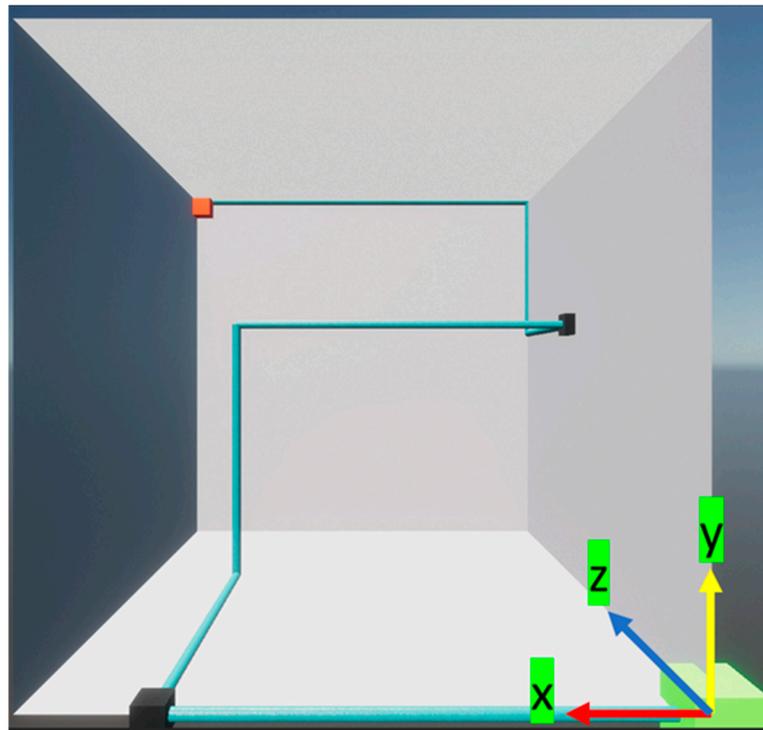


Figure 17. The layout of successful trial under constant condition for pass points task.

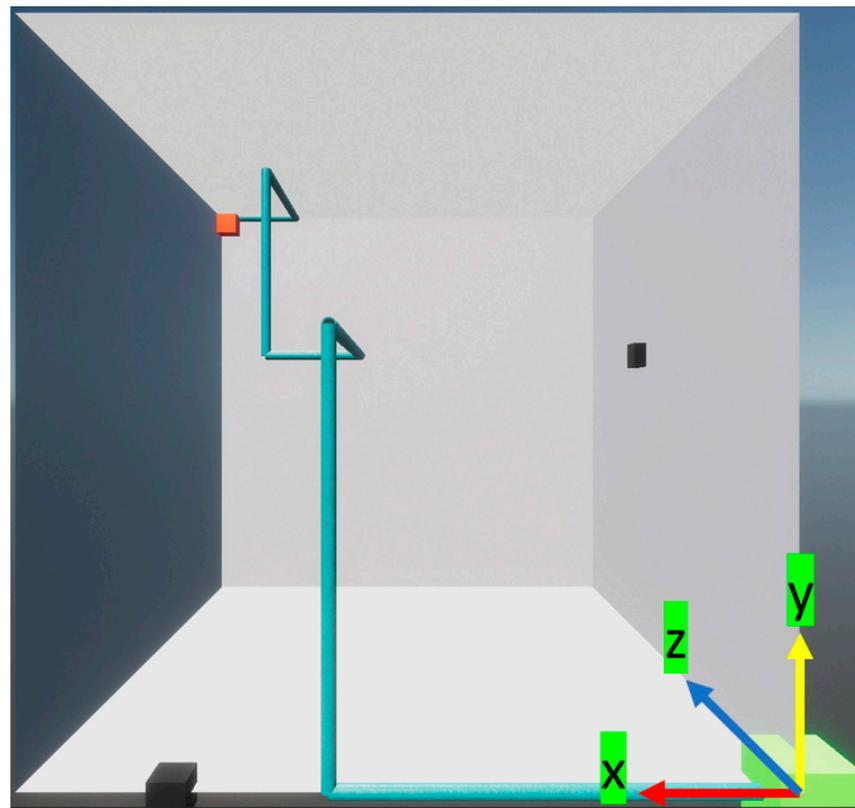


Figure 20. The layout of failed trial under variable conditions for pass points task.

Figures 17 and 18 show the successful and failed trials under constant conditions. The layout in Figure 18 further proves the shortcomings of ChatGPT in spatial understanding. There were two gaps along the pipeline, indicating that ChatGPT might wrongly overlap the two points only based on their 2D coordinates. The two end points in the red circle had the same x and z coordinates, but different y coordinates. The ones in the yellow circle had the same x and y coordinates, but different y coordinates. In other words, if the coordinates of the two points were the same along one or two axes, they would be wrongly aligned and treated as the same points. In this case, it is reasonable to deduce that ChatGPT relied more on pure, separated numerical analysis to solve the real-world problem. The x, y, and z coordinates of the two points might be separately compared and the two points would be considered as the same if the sum of the total difference is under a threshold. Even if the two end points were on the same x-z and z-y planes, they would still be treated as the same points in 3D space. Thus, the visual or multi-dimensional inputs are required for ChatGPT to build an accurate 3D scene for real-world operation. Figure 20 shows the influence of the constraint caused by using different sizes of pipe. The pipeline could only approach the mandatory points, but not pass them.

In conclusion, the system demonstrated superior performance under constant conditions in the second task as opposed to the first one. This can be attributed to the fact that avoiding specific points offered a greater array of potential solutions compared to passing points, resulting in a higher level of stability for the ChatGPT system. Consequently, the success rate for the second task was 1, whereas it was only 0.7 for the first task.

Considering the two tasks and the two conditions derived from real-world environments, it is evident that, in contrast to study cases 1 and 2, employing ChatGPT and RoboGPT systems to address real-world construction tasks introduces additional constraints that significantly impact the stability and overall performance of the system. Furthermore, it is crucial to recognize that addressing real-world tasks encompasses not only achieving the desired objectives, but also optimizing resource utilization. Consequently,

future research should aim to guide the ChatGPT agent towards identifying the most efficient and effective means of resolving the problem at hand.

6. Conclusions

In this paper, we presented a robotic system leveraging ChatGPT-4 for automated sequence planning in complex construction assembly tasks, such as assembling structural components of a building, installing electrical and plumbing systems, and coordinating the movement of construction equipment on site. The tasks involved a wide range of spatial constraints, including a limited workspace, safe operation distances, and the proper placement of components, as well as resource constraints, such as the availability of equipment and personnel. We developed a framework that allowed ChatGPT-4 to ingest relevant input data, including construction specifications, blueprints, and a list of available resources. The model was then able to generate an optimized assembly sequence plan by decomposing the tasks into logical steps, ensuring that the spatial and resource constraints were satisfied. Each step included specific instructions for the robotic system, such as the order of operations, the type and quantity of resources required, and the optimal path for the movement of equipment and materials. To evaluate the effectiveness of the ChatGPT-4-based method, we compared its performance with that of two real-world construction tasks. Our results showed that the ChatGPT-4-based system has the potential to understand the background logic of a sequential task and give a corresponding solution. We also used the test results from 80 trials to intuitively demonstrate the current limitations and boundaries of the ChatGPT agent in solving real-world tasks considering the physical constraints and resource restrictions. To be able to assist human workers in solving real construction problems, the abilities of ChatGPT for spatial understanding and dynamic management need to be improved.

Honestly, there are several limitations to our approach. First, we have yet to fully understand the underlying mechanisms that allow ChatGPT-4 to be used for construction task sequence planning, particularly when considering spatial and resource constraints. Second, the level of trust human workers have in the ChatGPT-4-based system remains unknown, which could impact the adoption of this technology in real-world scenarios. Lastly, ChatGPT-4's ability to process and analyze imagery data is limited, restricting its applicability in situations where visual information is crucial. Future research should focus on addressing these limitations and expanding the scope of the study. It is essential to test more construction applications to validate the robustness of the ChatGPT-4-based method and assess its performance across diverse tasks. Furthermore, investigating the reasons behind ChatGPT-4's success in construction task sequence planning will enhance our understanding of its capabilities and help improve the model. Additionally, integrating ChatGPT-4 with computer vision techniques could pave the way for a fully automated process, which would enable seamless collaboration between the language model and visual data processing systems, ultimately boosting efficiency and accuracy in construction sequence planning. Last but not least, in the pipeline installation task, we only used the successful rate as the performance indicator. However, multiple criteria should be considered in real construction tasks such as system robustness, different tasks' efficiency, and user satisfaction. The corresponding data should be collected and the multi-criteria methods should be applied to evaluate the overall performance of the system, including the stable preference ordering towards ideal solution (SPOTIS) [60] or RANKing COMparison (RANCOM) [61].

In our future work, we plan to augment our RoboGPT system with Reinforcement Learning from Human Feedback (RLHF) [62] to enhance its adaptability and robustness across a wide range of construction scenarios. To achieve this, we will design and integrate a feedback mechanism that enables the collection of human expert preferences and evaluations to guide the model's learning process. By incorporating RLHF, the RoboGPT system can iteratively update its sequence planning capabilities based on expert feedback, allowing it to better comprehend the intricacies and subtleties of construction tasks. This approach

enables the system to adapt more effectively to the dynamic nature of construction projects, while also reducing its reliance on large amounts of training data. Furthermore, we will develop a method for incorporating feedback from virtual simulations, which will reflect the consequences of the generated construction sequences. This additional source of feedback will enable the RoboGPT system to refine its calculations in real time and improve its overall performance. Also, as the current system serves as a research prototype, our future work will be directed towards enhancing its user interface. This is intended to increase its accessibility and utility for users with various levels of expertise and make the system more inclusive and user friendly for beginners.

Additionally, our experiments qualitatively illustrate that the use of RoboGPT can contribute to lower operational costs and increased efficiency across a range of construction tasks. Compared with the traditional robotic and automation methods, our proposed system could further reduce the need for human decision making and intervention and contribute to lower operational costs and increased efficiency. To make the system more applicable for general implementation, it is still worth quantitatively showing the actual increase in efficiency by using our system in specific tasks.

Author Contributions: Conceptualization, H.Y., Y.Y. and J.D.; methodology, H.Y., Y.Y. and J.D.; software, H.Y. and Y.Y.; validation, H.Y., T.Z. and Q.Z.; formal analysis, Q.Z. and J.D.; investigation, H.Y. and Y.Y.; resources, Y.Y.; writing—original draft preparation, H.Y. and J.D.; writing—review and editing, Q.Z. and J.D.; visualization, H.Y., T.Z. and Q.Z.; project administration, J.D.; funding acquisition, J.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Science Foundation (NSF), grant number 2128895. The APC was funded by the NSF.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author.

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

References

1. Ali, A.K.; Lee, O.J.; Song, H. Robot-based facade spatial assembly optimization. *J. Build. Eng.* **2021**, *33*, 101556. [[CrossRef](#)]
2. Zhu, A.; Xu, G.; Pauwels, P.; De Vries, B.; Fang, M. Deep Reinforcement Learning for Prefab Assembly Planning in Robot-based Prefabricated Construction. In Proceedings of the 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), Lyon, France, 23–27 August 2021; pp. 1282–1288.
3. Cai, S.; Ma, Z.; Skibniewski, M.J.; Bao, S. Construction automation and robotics for high-rise buildings over the past decades: A comprehensive review. *Adv. Eng. Inform.* **2019**, *42*, 100989. [[CrossRef](#)]
4. Liang, C.-J.; Kamat, V.R.; Menassa, C.C. Teaching robots to perform quasi-repetitive construction tasks through human demonstration. *Autom. Constr.* **2020**, *120*, 103370. [[CrossRef](#)]
5. Shi, Y.; Du, J.; Worthy, D.A. The impact of engineering information formats on learning and execution of construction operations: A virtual reality pipe maintenance experiment. *Autom. Constr.* **2020**, *119*, 103367. [[CrossRef](#)]
6. Abuwarda, Z.; Hegazy, T. Work-package planning and schedule optimization for projects with evolving constraints. *J. Comput. Civ. Eng.* **2016**, *30*, 04016022. [[CrossRef](#)]
7. Kim, T.; Kim, Y.-w.; Cho, H. Dynamic production scheduling model under due date uncertainty in precast concrete construction. *J. Clean. Prod.* **2020**, *257*, 120527. [[CrossRef](#)]
8. Garcia-Sabater, J.P.; Maheut, J.; Garcia-Sabater, J.J. A two-stage sequential planning scheme for integrated operations planning and scheduling system using MILP: The case of an engine assembler. *Flex. Serv. Manuf. J.* **2012**, *24*, 171–209. [[CrossRef](#)]
9. Suszyński, M.; Peta, K. Assembly sequence planning using artificial neural networks for mechanical parts based on selected criteria. *Appl. Sci.* **2021**, *11*, 10414. [[CrossRef](#)]
10. Lacave, C.; Diez, F.J. A review of explanation methods for heuristic expert systems. *Knowl. Eng. Rev.* **2004**, *19*, 133–146. [[CrossRef](#)]
11. L'heureux, A.; Grolinger, K.; Elyamany, H.F.; Capretz, M.A. Machine learning with big data: Challenges and approaches. *IEEE Access* **2017**, *5*, 7776–7797. [[CrossRef](#)]
12. Mitrović, S.; Andreoletti, D.; Ayoub, O. Chatgpt or human? detect and explain. explaining decisions of machine learning model for detecting short chatgpt-generated text. *arXiv* **2023**, arXiv:2301.13852.
13. Harinarain, N.; Caluza, S.; Dondolo, S. Bricklaying robots in the south african construction industry: The contractors perspective. In Proceedings of the Thirty-Seventh Annual Conference, Leeds, UK, 6–7 September 2021; p. 36.
14. Cardno, C.A. Robotic rebar-tying system uses artificial intelligence. *Civ. Eng. Mag. Arch.* **2018**, *88*, 38–39. [[CrossRef](#)]

15. Song, S.H.; Choi, J.O.; Lee, S. The Current State and Future Directions of Industrial Robotic Arms in Modular Construction. In Proceedings of the 9th International Conference on Construction Engineering and Project Management, Las Vegas, NV, USA, 20–23 June 2022; Korea Institute of Construction Engineering and Management: Seoul, Republic of Korea, 2022; pp. 336–343.
16. Apolinarska, A.A.; Pacher, M.; Li, H.; Cote, N.; Pastrana, R.; Gramazio, F.; Kohler, M. Robotic assembly of timber joints using reinforcement learning. *Autom. Constr.* **2021**, *125*, 103569. [[CrossRef](#)]
17. Gardner, L.; Kyvelou, P.; Herbert, G.; Buchanan, C. Testing and initial verification of the world’s first metal 3D printed bridge. *J. Constr. Steel Res.* **2020**, *172*, 106233. [[CrossRef](#)]
18. Oke, A.E.; Atofarati, J.O.; Bello, S.F. Awareness of 3D printing for sustainable construction in an emerging economy. *Constr. Econ. Build.* **2022**, *22*, 52–68. [[CrossRef](#)]
19. Gao, Y.; Meng, J.; Shu, J.; Liu, Y. BIM-based task and motion planning prototype for robotic assembly of COVID-19 hospitalisation light weight structures. *Autom. Constr.* **2022**, *140*, 104370. [[CrossRef](#)] [[PubMed](#)]
20. Carey, N.E.; Bardunias, P.; Nagpal, R.; Werfel, J. Validating a Termite-Inspired Construction Coordination Mechanism Using an Autonomous Robot. *Front. Robot. AI* **2021**, *8*, 645728. [[CrossRef](#)] [[PubMed](#)]
21. Ding, Y.; Dwivedi, R.; Kovacevic, R. Process planning for 8-axis robotized laser-based direct metal deposition system: A case on building revolved part. *Robot. Comput.-Integr. Manuf.* **2017**, *44*, 67–76. [[CrossRef](#)]
22. Zhang, J.; Shen, C.; Li, R. A Robotic System Method and Rebar Construction with Off-the-Shelf Robots. In *Computing in Civil Engineering 2021*; American Society of Civil Engineers: Reston, VA, USA, 2022; pp. 1204–1211.
23. Osa, T.; Aizawa, M. Deep reinforcement learning with adversarial training for automated excavation using depth images. *IEEE Access* **2022**, *10*, 4523–4535. [[CrossRef](#)]
24. Ardiny, H.; Witwicki, S.; Mondada, F. Construction automation with autonomous mobile robots: A review. In Proceedings of the 2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM), Tehran, Iran, 7–9 October 2015; pp. 418–424.
25. Adami, P.; Rodrigues, P.B.; Woods, P.J.; Becerik-Gerber, B.; Soibelman, L.; Copur-Gencturk, Y.; Lucas, G. Impact of VR-based training on human–robot interaction for remote operating construction robots. *J. Comput. Civ. Eng.* **2022**, *36*, 04022006. [[CrossRef](#)]
26. Alatartsev, S.; Stellmacher, S.; Ortmeier, F. Robotic task sequencing problem: A survey. *J. Intell. Robot. Syst.* **2015**, *80*, 279–298. [[CrossRef](#)]
27. Cao, T.; Sanderson, A.C. AND/OR net representation for robotic task sequence planning. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **1998**, *28*, 204–218. [[CrossRef](#)]
28. Xing, Y.; Chen, G.; Lai, X.; Jin, S.; Zhou, J. Assembly sequence planning of automobile body components based on liaison graph. *Assem. Autom.* **2007**, *27*, 157–164. [[CrossRef](#)]
29. Yassine, A. An introduction to modeling and analyzing complex product development processes using the design structure matrix (DSM) method. *Urbana* **2004**, *51*, 1–17.
30. Chen, I.-M.; Yang, G. Configuration independent kinematics for modular robots. In Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, MI, USA, 22–28 April 1996; pp. 1440–1445.
31. Craig, I. *Blackboard Systems*; Intellect Books: Bath, UK, 1995.
32. Chen, C.P.; Pao, Y.-H. An integration of neural network and rule-based systems for design and planning of mechanical assemblies. *IEEE Trans. Syst. Man Cybern.* **1993**, *23*, 1359–1371. [[CrossRef](#)]
33. Vossen, T.; Ball, M.; Lotem, A.; Nau, D. Applying integer programming to AI planning. *Knowl. Eng. Rev.* **2000**, *15*, 85–100. [[CrossRef](#)]
34. Sundström, N.; Wigström, O.; Falkman, P.; Lennartson, B. Optimization of operation sequences using constraint programming. *IEAC Proc. Vol.* **2012**, *45*, 1580–1585. [[CrossRef](#)]
35. Liu, Z.-F.; Hu, D.; Gao, X.; Zhang, J.-D. Product Disassembly Sequence Planning Based on Greedy Algorithm. *China Mech. Eng.* **2011**, *22*, 2162.
36. Shan, H.; Li, S.; Gong, D.; Lou, P. Genetic simulated annealing algorithm-based assembly sequence planning. In Proceedings of the 2006 International Technology and Innovation Conference (ITIC 2006), Stevenage, UK, 6–7 November 2006; pp. 1573–1579.
37. Li, W.; Ong, S.; Nee, A. Optimization of process plans using a constraint-based tabu search approach. *Int. J. Prod. Res.* **2004**, *42*, 1955–1985. [[CrossRef](#)]
38. Xia, H.; Li, X.; Gao, L. A hybrid genetic algorithm with variable neighborhood search for dynamic integrated process planning and scheduling. *Comput. Ind. Eng.* **2016**, *102*, 99–112. [[CrossRef](#)]
39. Berry, A.; Heggernes, P.; Simonet, G. The minimum degree heuristic and the minimal triangulation process. In Proceedings of the Graph-Theoretic Concepts in Computer Science: 29th International Workshop, WG 2003, Elspeet, The Netherlands, 19–21 June 2003; Revised Papers 29, 2003. pp. 58–70.
40. Yang, F.; Gao, K.; Simon, I.W.; Zhu, Y.; Su, R. Decomposition methods for manufacturing system scheduling: A survey. *IEEE/CAA J. Autom. Sin.* **2018**, *5*, 389–400. [[CrossRef](#)]
41. Tseng, H.-E.; Chang, C.-C.; Lee, S.-C.; Huang, Y.-M. A block-based genetic algorithm for disassembly sequence planning. *Expert Syst. Appl.* **2018**, *96*, 492–505. [[CrossRef](#)]
42. Han, Z.; Wang, Y.; Tian, D. Ant colony optimization for assembly sequence planning based on parameters optimization. *Front. Mech. Eng.* **2021**, *16*, 393–409. [[CrossRef](#)]
43. Bewoor, L.A.; Prakash, V.C.; Sapkal, S.U. Production scheduling optimization in foundry using hybrid Particle Swarm Optimization algorithm. *Procedia Manuf.* **2018**, *22*, 57–64. [[CrossRef](#)]

44. Arinez, J.F.; Chang, Q.; Gao, R.X.; Xu, C.; Zhang, J. Artificial intelligence in advanced manufacturing: Current status and future outlook. *J. Manuf. Sci. Eng.* **2020**, *142*, 110804. [[CrossRef](#)]
45. Strub, M.P.; Gammell, J.D. Adaptively Informed Trees (AIT): Fast Asymptotically Optimal Path Planning through Adaptive Heuristics. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 3191–3198.
46. Souza, M.J.; Coelho, I.M.; Ribas, S.; Santos, H.G.; Merschmann, L.H.d.C. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *Eur. J. Oper. Res.* **2010**, *207*, 1041–1051. [[CrossRef](#)]
47. Sun, H.; Burton, H.V.; Huang, H. Machine learning applications for building structural design and performance assessment: State-of-the-art review. *J. Build. Eng.* **2021**, *33*, 101816. [[CrossRef](#)]
48. Caiafa, C.F.; Sun, Z.; Tanaka, T.; Marti-Puig, P.; Solé-Casals, J. Machine learning methods with noisy, incomplete or small datasets. *Appl. Sci.* **2021**, *11*, 4132. [[CrossRef](#)]
49. Floridi, L.; Chiriatti, M. GPT-3: Its nature, scope, limits, and consequences. *Minds Mach.* **2020**, *30*, 681–694. [[CrossRef](#)]
50. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
51. Ciosici, M.R.; Derczynski, L. Training a T5 Using Lab-sized Resources. *arXiv* **2022**, arXiv:2208.12097.
52. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *arXiv* **2017**, arXiv:1706.03762.
53. Huang, W.; Xia, F.; Xiao, T.; Chan, H.; Liang, J.; Florence, P.; Zeng, A.; Tompson, J.; Mordatch, I.; Chebotar, Y. Inner monologue: Embodied reasoning through planning with language models. *arXiv* **2022**, arXiv:2207.05608.
54. Prieto, S.A.; Mengiste, E.T.; García de Soto, B. Investigating the use of ChatGPT for the scheduling of construction projects. *Buildings* **2023**, *13*, 857. [[CrossRef](#)]
55. Singh, I.; Blukis, V.; Mousavian, A.; Goyal, A.; Xu, D.; Tremblay, J.; Fox, D.; Thomason, J.; Garg, A. Progprompt: Generating situated robot task plans using large language models. *arXiv* **2022**, arXiv:2209.11302.
56. Ye, Y.; You, H.; Du, J. Improved trust in human-robot collaboration with ChatGPT. *IEEE Access* **2023**, *11*, 55748–55754. [[CrossRef](#)]
57. Vemprala, S.; Bonatti, R.; Bucker, A.; Kapoor, A. Chatgpt for robotics: Design principles and model abilities. *Microsoft Auton. Syst. Robot. Res.* **2023**, *2*, 20.
58. Bang, Y.; Cahyawijaya, S.; Lee, N.; Dai, W.; Su, D.; Wilie, B.; Lovenia, H.; Ji, Z.; Yu, T.; Chung, W. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv* **2023**, arXiv:2302.04023.
59. Badini, S.; Regondi, S.; Frontoni, E.; Pugliese, R. Assessing the capabilities of ChatGPT to improve additive manufacturing troubleshooting. *Adv. Ind. Eng. Polym. Res.* **2023**, *6*, 278–287. [[CrossRef](#)]
60. Dezert, J.; Tchamova, A.; Han, D.; Tacnet, J.-M. The SPOTIS rank reversal free method for multi-criteria decision-making support. In Proceedings of the 2020 IEEE 23rd International Conference on Information Fusion (FUSION), Rustenburg, South Africa, 6–9 July 2020; pp. 1–8. [[CrossRef](#)]
61. Więckowski, J.; Kizielewicz, B.; Shekhovtsov, A.; Sałabun, W. RANCOM: A novel approach to identifying criteria relevance based on inaccuracy expert judgments. *Eng. Appl. Artif. Intell.* **2023**, *122*, 106114. [[CrossRef](#)]
62. Christiano, P.F.; Leike, J.; Brown, T.; Martic, M.; Legg, S.; Amodei, D. Deep reinforcement learning from human preferences. *arXiv* **2017**, arXiv:1706.03741.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.