Democratizing Construction Robotics: A Neuro-Symbolic Framework for Semantic Spatial Adaptation and Intuitive Control

Hui-Yu (Leon) Lin[1],Liang-Ting (Tim) Tsai[2†],Cheng-Hsuan (Jason) Yang[3†],Tzong-Hann Wu[1†], Shang-Hsien (Patrick) Hsieh[1*]

[1]Department of Civil Engineering, National Taiwan University, Taiwan
[2]Department of Civil and Environmental Engineering, University of Alberta, Canada
[3]RoBIM Technologies Inc, Canada
[†]These authors contributed equally as second authors

*Corresponding author shhsieh@ntu.edu.tw

Abstract -

This paper presents a neuro-symbolic framework for intuitive robot control in construction, addressing the technical complexity barrier that limits robotics adoption on construction sites. The framework comprises two components: a Neural Component (large language model) that interprets natural language instructions and generates high-level spatial primitives, and a Symbolic Component that transforms these primitives into precise robot coordinates through deterministic computation. This separation ensures that the language model cannot produce invalid robot motions, as all outputs are validated through geometric calculation. The framework was evaluated through proof-of-concept experiments involving brick wall stacking (9 objects, 54 motions) and timber beam arrangement (5 objects, 30 motions) in a physics-based simulation. Results demonstrated over 90% reduction in task specification time compared to manual coordinate entry. A human-in-the-loop verification process enables users to validate planned motions through 3D visualization and refine tasks through iterative natural language dialogue, typically converging within 3-5 conversation rounds. The framework lowers the barrier to robot programming by enabling construction practitioners to express spatial intent using domain-familiar vocabulary rather than coordinates or code.

Keywords -

Construction Automation, Large Language Models (LLM), Human-Robot Collaboration, Natural Language Interface

1 Introduction

The global construction industry faces a serious worker shortage. According to the Associated Builders and Contractors, approximately 70% of construction firms struggle to find enough workers [1], with the average age of craft workers projected to reach 46 by 2030. Meanwhile, the construction robotics market is projected to grow from $5.5 billion in 2024 to $12.99 billion by 2029 [2], showing strong industry interest in automation.

Despite this strong market growth and industry interest, the practical adoption of construction robotics remains limited. Prior studies suggest that barriers extend well beyond initial investment costs. Tafazzoli et al. [3] surveyed construction stakeholders and found that "technical complexity" and "lack of skilled labor" are major barriers. Liu et al. [4] reviewed 95 articles and concluded that "insufficient training and knowledge" limits adoption. Prieto et al. [5] noted that construction practitioners "without particular expertise in robotics" struggle to use current systems because they require "high expertise for robotic knowledge."

This creates a problem: engineers who know what needs to be built often cannot tell robots how to do it. Traditional industrial robots need exact instructions written in specialized programming languages, which most construction workers do not know.

While construction robotics encompasses various platforms including drones for site surveying, mobile robots for material transport, and exoskeletons for worker augmentation, this research focuses specifically on industrial manipulators (robotic arms) used for material handling and assembly tasks. These manipulators present the most significant programming barrier because they require precise spatial coordinates for every motion, unlike platforms that can operate with higher-level navigation commands or follow predefined paths.

To solve this problem, this research proposes a framework with two main parts. The first part is a Large Language Model (LLM) that reads natural language instructions and converts them into simple spatial commands like "move left" or "place in a grid." The second part is a calculation module that takes the commands and computes the exact positions the robot needs to move to.

The LLM only outputs simple commands, not exact

coordinates. This means that even if the LLM makes a mistake, the calculation module will still produce valid robot movements. The user can then watch a 3D preview and ask for changes using plain language.

The contribution of this work is not robot path planning or motion generation, but rather human-to-robot communication: enabling construction workers to express spatial intent using familiar language ("left," "forward," "in a grid") rather than coordinates or code.

2 Related work

2.1 Barriers to using robots in construction

Several studies have found that robots are hard to use in construction. Tafazzoli et al. [3] surveyed construction companies and found "technical complexity" and "lack of skilled workers" as top problems. Their survey revealed that construction professionals see current robotic systems as requiring expertise they do not have. Liu et al. [4] reviewed 95 papers spanning 1974-2023 and concluded that "not enough training and knowledge" limits robot use. They noted that current technology requires "expertise in robotics" that most construction workers lack. Li et al. [6] reviewed human-robot teamwork in construction from 2014-2024 and found a trend toward making systems easier for humans to use. They documented barriers across human, robotic, and interaction domains. These findings show why we need simpler robot interfaces rather than more capable ones.

2.2 Human-robot interaction and digital integration in construction

Various interaction paradigms have been explored for controlling robots in construction and related domains. Vision-based approaches use cameras and gesture recognition to interpret worker intent, though they can be affected by occlusion and lighting conditions common on construction sites. Wearable sensing through IMUs or EMG sensors enables intuitive motion-based control but requires workers to don additional equipment. Voice-based interfaces offer hands-free operation, which is advantageous when workers are handling materials or tools. The proposed framework adopts a natural language approach that extends beyond simple voice commands by leveraging large language models to interpret complex spatial instructions expressed in workers' own vocabulary.

Complementing these interaction methods, research on digital integration has explored connecting Building Information Models (BIM) to robots. Zhu et al. [16] created a way to convert BIM files (IFC format) into robot simulation formats, enabling ROS-based robot simulation directly from BIM models. Kim and Peavy [17] proposed BIM-based semantic building world modeling with two-way information flow between robots and digital twins. Recent review studies on BIM-robotics integration have reported that IFC-based data exchanges primarily focus on basic geometric information while lacking rich semantic representations required for higher-level robot reasoning, highlighting the need for an additional semantic reasoning layer [18]. Karimi et al. [19] proposed the Building Information Robotic System (BIRS), which converts IFC data into graphs for semantic path planning, considering construction activities and hazard zones.

Du et al. [11] combined language models with BIM data, using IfcOpenShell for data extraction and reducing task time by 60%. Their system reads building data efficiently for tasks like bricklaying and 3D concrete printing. While these BIM-based approaches provide valuable geometric data, they still require users to understand structured data formats and do not directly address the usability barrier. The proposed framework focuses on a complementary problem: understanding what users say using relative, human-centric instructions ("left," "forward") rather than absolute coordinates. Future integration with BIM could combine the strengths of both approaches (discussed in Section 5.3).

2.3 Using language models to control robots

To make robot control more intuitive, recent research has explored using language models. Liang et al. [7] showed that language models can write Python code to control robot arms, demonstrating that LLM-generated policies can handle spatial-geometric reasoning and generalize to new instructions. Huang et al. [8] created VoxPoser, which achieved 76.7% success on new instructions by creating 3D maps of where robots should move. Wang et al. [9] combined language models with motion planning, using GPT-4 as task planner with iterative refinement through motion feedback. Park et al. [10] demonstrated 99% accuracy for natural language robot control in drywall installation, showing the potential of this approach for construction. However, these systems are designed for robotics experts. The proposed framework adapts these ideas for construction workers who are not programmers.

2.4 Combining language models with calculations

Since language models can sometimes make mistakes, combining them with calculation modules (exact but rigid) is a growing research area. Capitanelli and Mastrogiovanni [12] showed that this combination achieves 95.5% success on planning tasks while generating plans 13.5% shorter than pure symbolic approaches. Rana et al. [13] used 3D scene graphs to help language models plan better, reducing input size by 60-82% while maintaining accuracy.

Gu et al. [14] created ConceptGraphs, which builds 3D maps from images with semantic understanding, enabling LLM-based planning through natural text queries. Chen et al. [15] developed SpatialVLM, showing that vision-language models can answer spatial queries like "is A to the left of B?" These works show that combining language understanding with exact calculations works well, which supports our approach.

3 Methodology

3.1 System overview

Figure 1 shows the overall system. The proposed framework has three major components: the Neural Component, the Symbolic Component, and the Execution Layer. The Neural Component is a language model that interprets natural language input and outputs structured spatial commands. The Symbolic Component is a calculation module that converts these commands into exact robot positions. The Execution Layer handles safety checks and generates executable robot trajectories.

The process works as follows: (1) the user gives an instruction like "stack bricks in a 3 by 3 grid"; (2) the Neural Component reads this and outputs simple spatial commands; (3) the Symbolic Component calculates exact positions from these commands; (4) the Execution Layer checks if the movements are safe; (5) a Human-in-the-Loop verification step allows the user to preview the planned motion in 3D simulation and provide feedback; and (6) once approved, the robot executes the motion.

The key design choice is that the Neural Component only outputs simple commands like "GRID" or "LEFT," not exact coordinates. This means even if the language model misunderstands something, the Symbolic Component will still produce valid robot positions. The Human-in-the-Loop step ensures that users can catch any errors before physical execution.

3.2 Neural component

The Neural Component is the language model that reads what the user says. It does three things. First, it figures out what task the user wants (like pick-and-place or arranging items). Second, it understands which objects the user means (like "the table" means Table 1). Third, it picks which spatial command to use (like GRID for grid placement or OFFSET for moving something).

The framework can use any language model (GPT-4, Claude, Gemini, or others). The Neural Component outputs simple data like: {`action: OFFSET, ref: Table_1, dir: LEFT, mag: 0.1`}. Because it only outputs these simple commands, it cannot produce impossible robot movements.

What the Neural Component Does and Does Not Do. The boundary of the Neural Component's responsibility is intentionally limited. The Neural Component does: interpret natural language, recognize user intent, resolve object references, and select appropriate spatial primitives. The Neural Component does not: calculate coordinates, check for collisions, solve inverse kinematics, or generate trajectories. These tasks are handled by the Symbolic Component and Execution Layer, which are deterministic and verifiable. This separation is a deliberate design choice.

Interaction Primitives as Domain Vocabulary. The spatial commands (LEFT, GRID, OFFSET, ALIGN, WALL, LINEAR) are not merely implementation details. They constitute a domain-native interaction vocabulary designed to reflect how construction workers naturally communicate spatial intent. When a worker says "stack them in a grid" or "move it left," these expressions map directly to primitives that the system understands. This vocabulary bridges the gap between worksite language and robot coordinate systems.

3.3 Symbolic component

The Symbolic Component is the calculation module that does the mathematical computations. It converts object names to actual positions in space. It calculates exact coordinates using the commands from the Neural Component. It also checks that positions are within the robot's reach.

This component always gives the same answer for the same input. There is no guessing involved.

The position calculation uses this formula:

$$P_{new} = P_{old} + R_{ref} \cdot \vec{v}_{semantic} \tag{1}$$

where R_{ref} is the rotation of the reference object, and $\vec{v}_{semantic}$ comes from the spatial command (for example, "Left" becomes $[-d, 0, 0]^T$ where d is the distance).

When a user says "left of the table," the system uses the table's direction, not the robot's. This matches how people naturally think about directions. Chen et al. [15] showed that vision-language models can understand such spatial relationships.

A key advantage of this approach is support for relative spatial references. When a user says "starting from the left edge," the system resolves positions relative to the target object's local coordinate frame, eliminating the need to calibrate the robot against global site coordinates. This is particularly valuable for mobile manipulators that relocate between work stations.

3.4 Execution layer

The Execution Layer is the final safety check. It calculates how each robot joint should move. It checks for

collisions with objects. It also creates smooth movement paths.

If something is wrong (like the target is too far), the system tells the user in plain words ("the robot cannot reach that position") instead of showing error codes.

3.5 Human-in-the-loop verification

To enable broader adoption, construction practitioners should be able to verify robot behavior without requiring programming or robotics expertise. Visual validation mechanisms are therefore essential for supporting decision-making and error detection. In this study, a three-dimensional preview is introduced to allow users to visually inspect and assess robot actions prior to execution, as shown in Figure 2.

The verification process works as follows. First, the system creates a 3D animation showing what the robot will do. Second, the user watches the animation. Third, the user decides if it looks correct based on their construction knowledge ("Does the brick placement look right?"). Fourth, if something needs to change, the user provides feedback in plain language ("move the grid 10cm backward").

This approach ensures that users only need construction domain knowledge, not programming skills. Users observe the virtual robot motion and apply their professional judgment. The time required for visual verification and adjustment is approximately 1 minute, compared to 25-45 minutes for manual coordinate specification.

For tasks that repeat, the position of item i is calculated as:

$$P_{pick}(i) = P_{pick}(0) + i \cdot \Delta_{offset} \quad (2)$$

where Δ_{offset} comes from the user's instruction (such as "10cm to the left each time").

4 Proof-of-concept

To demonstrate the feasibility of the proposed framework, two scenarios were tested in simulation: brick wall stacking and timber beam arrangement. These scenarios represent common construction tasks that involve repetitive pick-and-place operations.

4.1 Experimental setup

The experiments were conducted using a simulated KUKA KR60 industrial robot in a physics-based simulation environment. The simulation included collision detection, inverse kinematics solving, and 3D visualization for verification. The Neural Component used Gemini 2.0 Flash (model identifier: gemini-2.0-flash) as the language model, accessed through the Google AI API with temperature set to 0.2 for deterministic outputs. The structured prompt (approximately 800 tokens) defined the available spatial commands (WALL, LINEAR, OFFSET, ALIGN), reference frame conventions, and expected JSON output schema. While Gemini 2.0 Flash was used for this proof-of-concept, the framework architecture is model-agnostic and can accommodate other LLMs (GPT-4, Claude, Llama, etc.) by adapting the prompt format.

One of the developers served as the participant for the experiment. The participant has a background in civil engineering and robotics simulation, but had no industrial robot programming experience prior to this research. The participant conducted 5 trials of motion planning for each scenario, for a total of 10 trials. For each trial, the time required to specify the task was recorded using a stopwatch, and the number of conversation rounds needed to achieve the desired result was counted.

To measure the manual specification baseline, the participant entered all coordinates directly into the robot programming interface without using the proposed framework. This process involved: looking up object positions from the simulation scene, calculating layer and position offsets manually using a calculator, entering each coordinate into the interface, and setting motion parameters (speed, acceleration, gripper timing). The manual specification time reported in Table 2 represents the average across 3 baseline trials.

The current implementation uses English for all interactions. The example instructions shown below are written clearly for demonstration purposes. In practice, construction workers may express the same intent differently, such as "stack them up over there" or "make a wall with these bricks." The use of a language model rather than a rule-based parser allows the system to handle such variations in phrasing.

4.2 Scenario 1: Brick wall stacking

The task required programming the robot to build a brick wall by picking bricks from a supply table (Table 1) and stacking them on a work surface (Table 2). The wall consists of 3 layers with 3 bricks per layer, for a total of 9 bricks. Without the proposed framework, this task would require writing robot programming code specifying exact coordinates for 9 pick positions and 9 place positions at different heights.

The natural language input was: "Pick bricks from Table 1 and stack them as a wall on Table 2 with 3 layers, 3 bricks per layer."

The Neural Component output the following spatial commands:

```
{task: PICK_PLACE_WALL, source: Table_1,
 target: Table_2, layers: 3,
 bricks_per_layer: 3, approach: Z_DOWN}
```

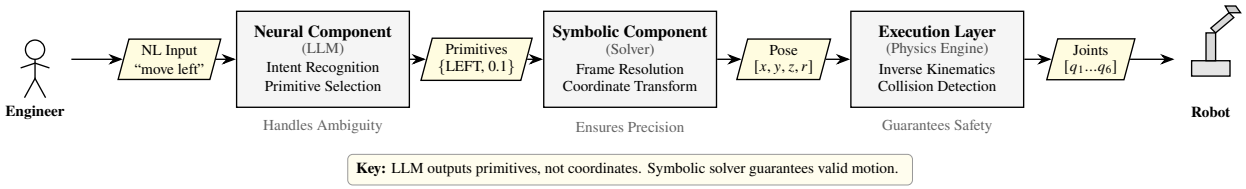

Figure 1. System architecture.

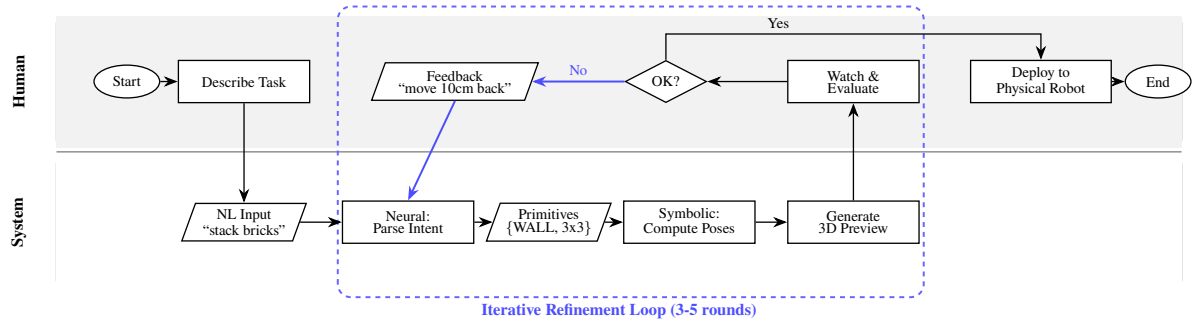

Figure 2. Human-in-the-loop workflow.

The Symbolic Component expanded this into 54 Cartesian poses (6 motions per brick multiplied by 9 bricks). For each brick at layer l and position p within that layer:

$$P_{place}(l, p) = P_{base} + \begin{bmatrix} p \cdot \Delta_x \\ 0 \\ l \cdot \Delta_z \end{bmatrix} \tag{3}$$

where P_{base} is the starting corner of Table 2, Δ_x is the brick width, and Δ_z is the brick height.

During simulation preview, the participant observed that the wall was too close to the table edge. The feedback was: “Move the wall 10cm backward.” The system applied a global offset to all 9 place positions without requiring coordinate recalculation.

4.3 Scenario 2: Timber beam arrangement

The second scenario involved arranging timber beams in a row for a prefabricated wall frame, requiring consistent spacing and parallel alignment. Beams were stored on Stack_1 and placed on Assembly_Table. This pattern is common in timber frame construction and modular assembly.

The natural language input was: “Place 5 timber beams from Stack_1 onto Assembly_Table. Line them up parallel, evenly spaced, starting from the left edge.”

The Neural Component output the following abstract primitives:

```
{task: PICK_PLACE_LINEAR, source: Stack_1,
 target: Assembly_Table, count: 5,
 spacing: EVEN, alignment: PARALLEL,
 start: LEFT_EDGE}
```

Table 1. Motion cycle structure (per object).

#	Action	Gripper
1	Approach (above pick)	Open
2	Descend (to surface)	Open
3	Grasp	Close
4	Lift	Close
5	Transfer (to place)	Close
6	Release	Open

The Symbolic Component generated 30 poses (6 motions per beam multiplied by 5 beams), computing positions relative to Assembly_Table’s left edge coordinate using linear interpolation.

As a refinement example, the participant said: “Rotate all beams 90 degrees.” The system updated orientation parameters globally.

4.4 Results and metrics

Table 1 shows the 6-motion cycle structure common to both scenarios. Each pick-and-place operation follows this sequence: approach the pick position, descend to object surface, grasp, lift, transfer to place position, and release.

Table 2 summarizes quantitative metrics across both scenarios. The manual specification time (40 minutes for wall stacking, 25 minutes for timber arrangement) represents the average across baseline trials. The manual process involved entering all coordinates directly into the robot programming interface, including: looking up object positions from the scene, calculating layer and position offsets manually, entering each coordinate, and setting motion parameters (speed, acceleration, gripper timing).

Table 2. Proof-of-concept metrics and time comparison.

Metric	Wall	Timber
Motions generated	54	30
Generation time	8 sec	5 sec
Preview duration	45 sec	30 sec
Manual specification	40 min	25 min
Proposed framework	3 min	2 min
Time reduction	**92%**	**92%**

Table 3. Comparison with other LLM-robot systems.

System	LLM Output	Calculation	Target User
Code as Policies	Code	Minimal	Roboticist
VoxPoser	Code + maps	Partial	Roboticist
Du et al.	Tool calls	IFC-based	BIM expert
Proposed	**Primitives**	**Full**	**Construction**

The proposed framework reduces this to approximately 3 minutes (wall) and 2 minutes (timber), including natural language input, Neural Component processing, preview watching, and one refinement iteration.

Accuracy and Reliability. The Neural Component correctly interpreted task intent in all 10 trials within this limited evaluation. The Symbolic Component is fully deterministic, and because the Neural Component outputs only high-level primitives rather than raw coordinates, the risk of LLM hallucination producing invalid robot positions is significantly mitigated by design.

Physical manipulation reliability (grasping precision, placement accuracy) was not measured in this simulation-based study, as the research focus is on the human-robot communication interface. While the current evaluation is limited in scale, the results demonstrate the feasibility of the approach. Future work will include larger-scale benchmarks and physical validation to further assess robustness.

4.5 Usability observations

During testing, the participant was able to generate the desired motion sequences within 3 to 5 rounds of conversation with the system. A typical interaction pattern was: (1) initial task description, (2) watching the 3D preview, (3) one or two refinement requests, and (4) confirmation.

The refinements fell into two categories: spatial adjustments (changing positions or orientations) and linguistic clarifications (resolving interpretation ambiguity). Approximately half were user-requested spatial adjustments, and half were corrections due to linguistic misunderstanding, suggesting that improved prompting and spatial preview tools could reduce the number of rounds needed.

4.6 Failure case and recovery

To illustrate error handling, a failure case from testing is documented. The participant instructed: "move the bricks slightly to the left." The Neural Component interpreted "left" relative to the robot base frame rather than the table frame, resulting in diagonal displacement visible in the simulation preview.

The participant immediately noticed the incorrect motion direction during 3D preview without any code inspection. The clarification was: "I meant left relative to the table, not the robot." The Neural Component correctly re-interpreted the instruction using the table's local coordinate frame, and the Symbolic Component regenerated all affected poses.

This case shows the importance of visual verification for catching misinterpretation, and the ability to recover through natural language clarification rather than code debugging.

5 Discussion

The proposed framework was designed with a specific goal: making robot programming accessible to construction workers who have no coding experience. This section examines how the framework compares to existing approaches, what it contributes to the field, and where it can be improved.

5.1 Comparison with other systems

Several research groups have explored using LLMs to control robots, but most systems target robotics experts rather than end users. Table 3 compares the proposed framework with representative systems.

Other systems are designed for robotics experts or BIM specialists. The proposed framework is designed for construction workers who know their job but cannot write code. Users check results by watching 3D animations, not by reading code.

Compared to Du et al. [11], the two systems work differently. Du et al.'s users give exact coordinates ("place brick at [x,y,z]"). Users of the proposed framework give relative instructions ("move left") without numbers. Du et al. focus on reading BIM files efficiently. The proposed framework focuses on understanding natural language. The two approaches could work together: use the proposed interface for communication and Du et al.'s system for reading BIM data.

5.2 Contributions

The experiments demonstrated two main contributions of this work:

1. **Lowering the Robot Programming Barrier.** The experiments show that language-based interfaces can change robot programming from writing code to having a conversation. This directly addresses the problems found by Tafazzoli et al. [3] and Liu et al. [4]: technical complexity and lack of training. Table 2 shows over 90% time reduction for both tasks. More importantly, users no longer need to understand robot coordinates. They just need to describe tasks in their own words, which construction workers already know how to do.
2. **Reducing Interaction Complexity through LLM.** If the framework only outputs simple commands, why not just use keyword matching? The answer is that people say the same thing in many different ways. "Move it left," "shift leftward," and "push it toward the west" should all do the same thing. A rule-based system would need to list every possible phrase. An LLM can understand new phrases it has never seen before. This flexibility is why a language model is necessary.

5.3 Future work

While the proof-of-concept results are promising, the current framework has several limitations that must be addressed before deployment in real construction environments. This section discusses these limitations and outlines directions for future development.

1. **Third-Party User Testing.** The current evaluation was conducted only by one of the developers, who understands exactly how the system works. To ensure the interface is truly intuitive for construction workers, future studies should recruit participants with no robotics background and measure their success rates and learning curves.
2. **Complex Object Representation.** Construction sites contain objects with complex shapes that cannot be represented as simple bounding boxes. Future versions should support mesh-based representations and collision detection for irregular geometries such as pipes, ducts, and structural connections.
3. **Expanding Spatial Primitives.** The current set of spatial commands (WALL, LINEAR, OFFSET, ALIGN) covers common repetitive pick-and-place tasks but cannot handle all construction scenarios. Complex assembly sequences, curved arrangements, or tasks requiring conditional logic would need additional primitives. Future work should investigate extensible primitive libraries that can be customized for specific construction trades and task types.
4. **Automatic Scene Specification and Drawing Integration.** The proposed framework is not intended to replace drawing-based workflows, but to complement them in scenarios where process-level, ad-hoc, or intermediate tasks are not explicitly defined in design documents. Integrating with BIM through IFC-to-Scene mapping could allow automatic parameter extraction. However, drawings typically represent final states rather than intermediate process tasks, and calibrating robot coordinates to global BIM coordinates remains challenging for mobile work stations. A promising direction is hybrid approaches where BIM data provides default parameters that workers can override through natural language.
5. **Physical Deployment.** Real construction environments introduce challenges that simulation cannot capture, including sensor noise, lighting variations, and mechanical tolerances. Future work should incorporate visual servoing and force feedback to handle the gap between planned and actual positions.
6. **Reducing Refinement Rounds.** Users currently need 3 to 5 conversation rounds to achieve their desired result. Improving the LLM prompts and adding spatial disambiguation features could reduce this number, making the system faster to use.
7. **Enhanced Safety Mitigations.** While the current system includes basic safety checks, industrial deployment would require more sophisticated mechanisms such as real-time obstacle detection, speed limiting near humans, and certified emergency stop protocols.

6 Conclusion

This research presents a framework that enables construction practitioners to interact with and control robotic systems through natural language. Rather than generating low-level motion parameters or explicit coordinates, the Neural Component produces high-level, task-oriented commands, thereby reducing the risk of infeasible or unsafe robot motions. The generated actions are validated through 3D simulation and animation, allowing users to verify outcomes visually instead of interpreting source code. Furthermore, task refinements and corrections are achieved through iterative natural language interaction, eliminating the need for direct code modification.

The experiments demonstrated over 90% reduction in the time required to set up robot tasks compared to manual coordinate specification. Users only need construction domain knowledge, not programming expertise.

As the construction industry faces worker shortages and increasing interest in automation, the complexity of robot programming remains a significant barrier to adoption. This work aims to make robot control as intuitive as giving instructions to a human worker, by separating natural language understanding from geometric computation. This

approach lowers the technical barrier to robot programming and supports more intuitive, transparent, and user-centered human-robot interaction in construction contexts.

Acknowledgments

The authors gratefully acknowledge the support of the industry-academia collaboration project between National Taiwan University and RoBIM Technologies Inc., which made this research possible.

References

[1] Associated Builders and Contractors, "Construction Workforce Shortage Report," 2024. Accessed: 01/12/2025.

[2] MarketsandMarkets, "Construction Robots Market Report," 2024. Accessed: 01/12/2025.

[3] M. Tafazzoli, K. Shrestha, and H. Dang, "Investigating Barriers to the Application of Automation in the Construction Industry," in *Proc. ASCE Construction Research Congress*, pp. 941-950, 2024. doi: 10.1061/9780784485262.096.

[4] Y. Liu, A.H. Alias, N.A. Haron, M.N.A. Bakar, and H. Wang, "Robotics in the Construction Sector: Trends, Advances, and Challenges," *Journal of Intelligent & Robotic Systems*, vol. 110, no. 72, 2024. doi: 10.1007/s10846-024-02104-4.

[5] S.A. Prieto, X. Xu, and B. García de Soto, "A guide for construction practitioners to integrate robotic systems in their construction applications," *Frontiers in Built Environment*, vol. 10, 1307728, 2024. doi: 10.3389/fbuil.2024.1307728.

[6] Z. Li et al., "Human-Centric Human-Robot Collaboration in On-Site Construction (2014-2024): Advances, Barriers, and Future Directions," *Automation in Construction*, 2025. doi: 10.1016/j.autcon.2025.106566.

[7] J. Liang, W. Huang, F. Xia, et al., "Code as Policies: Language Model Programs for Embodied Control," in *Proc. IEEE ICRA*, pp. 9493-9500, 2023. doi: 10.1109/ICRA48891.2023.10160557.

[8] W. Huang, C. Wang, R. Zhang, et al., "VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models," in *Proc. CoRL*, PMLR vol. 229, 2023. doi: 10.48550/arXiv.2307.05973.

[9] S. Wang, M. Han, Z. Jiao, et al., "LLM^3: Large Language Model-based Task and Motion Planning with Motion Failure Reasoning," in *Proc. IEEE/RSJ IROS*, 2024. doi: 10.48550/arXiv.2403.11144.

[10] S. Park, X. Wang, C.C. Menassa, V.R. Kamat, and J.Y. Chai, "Natural Language Instructions for Intuitive Human Interaction with Robotic Assistants in Field Construction Work," *Automation in Construction*, vol. 161, 105345, 2024. doi: 10.1016/j.autcon.2024.105345.

[11] S. Du, Y. Gao, W. Tong, and Y. Weng, "Towards Agent: LLM-based Framework for Robotic Construction by Leveraging IFC," in *Proc. 42nd International Symposium on Automation and Robotics in Construction (ISARC)*, pp. 358-365, 2025. doi: 10.22260/ISARC2025/0046.

[12] A. Capitanelli and F. Mastrogiovanni, "A Framework for Neurosymbolic Robot Action Planning using Large Language Models," *Frontiers in Neurorobotics*, vol. 18, 1342786, 2024. doi: 10.3389/fnbot.2024.1342786.

[13] K. Rana, J. Haviland, S. Garg, et al., "SayPlan: Grounding Large Language Models using 3D Scene Graphs for Scalable Robot Task Planning," in *Proc. CoRL*, PMLR vol. 229, pp. 23-72, 2023. doi: 10.48550/arXiv.2307.03668.

[14] Q. Gu, A. Kuwajerwala, S. Morin, et al., "ConceptGraphs: Open-Vocabulary 3D Scene Graphs for Perception and Planning," in *Proc. IEEE ICRA*, pp. 5021-5028, 2024. doi: 10.1109/ICRA57147.2024.10610332.

[15] B. Chen et al., "SpatialVLM: Endowing Vision-Language Models with Spatial Reasoning Capabilities," in *Proc. IEEE/CVF CVPR*, 2024. doi: 10.1109/CVPR52733.2024.01323.

[16] A. Zhu, P. Pauwels, and B. De Vries, "Component-based robot prefabricated construction simulation using IFC-based building information models," *Automation in Construction*, vol. 152, 104899, 2023. doi: 10.1016/j.autcon.2023.104899.

[17] K. Kim and M. Peavy, "BIM-based Semantic Building World Modeling for Robot Task Planning and Execution in Built Environments," *Automation in Construction*, vol. 138, 104247, 2022. doi: 10.1016/j.autcon.2022.104247.

[18] O. Odugu, F. Ghafari, E. Shourangiz, M. T. Khan, and C. Wang, "BIM-driven robotics in construction: A systematic review of integration methods, applications, and future directions," *Journal of Intelligent Construction*, 2025. doi: 10.26599/JIC.2025.9180103.

[19] S. Karimi, I. Iordanova, and D. St-Onge, "An ontology-based approach to data exchanges for robot navigation on construction sites," *Journal of Information Technology in Construction (ITcon)*, vol. 26, 2021. doi: 10.36680/j.itcon.2021.026.