AHFE 2025: Vol. XX, 2024
doi: 10.54941/ahfeXXXX

Human Language-Instructed Robotic Excavation based on Behavior Trees

Zirui Hong[1] and Hubo Cai[1]

[1]School of Civil Engineering, Purdue University, West Lafayette, IN 47907, USA

ABSTRACT

Collaborative construction robots have emerged as a promising alternative to relieve construction workers from both physically and cognitively demanding tasks, contributing to a safer and more productive construction industry. However, communicating with robots is not a trivial task as human workers and robots speak different languages. From the human-centered perspective, allowing human workers to communicate with robots using natural language is desirable because it minimizes additional cognitive load to human workers. Existing studies, however, have been focusing on converting language instructions into sequential actions, leading to a rigid task plan and inability to handle complex situations and unstructured working environments. To address this critical limitation, this paper explores the use of behavior tree (BT), an alternative architecture for describing and controlling complex tasks like excavation. A behavior tree is a hierarchical tree structure that specifies the switching between the agent's actions (i.e., execution nodes) via control flow nodes. Its modular nature allows the BT of excavation to be generated through linking reusable actions based on the human task descriptions. The resulting BT structure enables the robot to alter its behavior by selecting different tree branches in response to changing working conditions, thus improving its adaptability to dynamic construction environment and its capability of error-handling. In addition, the BT eases the human understanding of robot behavior for debugging and correcting robot behavior. A corresponding framework is proposed for enabling humans to guide a robotic excavator using goal-oriented language instructions. The framework consists of four modules: interpretation and reasoning, knowledge management, structural analysis and parsing, and BT generation. The interpretation and reasoning module decomposes instructions into structured executable intents. The knowledge management module organizes the knowledge for instruction reasoning, including the robot capable skills and its current working environment. Structure analysis and parsing module further grounds the intents and extracts associated parameters, while BT generation module maps the extracted elements with predefined BT nodes, building and refining the BTs of desired tasks. A case illustration is performed to demonstrate the viability of the proposed framework with executable demos. The findings are expected to facilitate efficient and transparent human-robot cooperation in earthmoving construction from a human friendly perspective.

Keywords: Human-robot collaboration, Human-robot communication, Natural language understanding, Robot task planning and control, Human-centered design, Behavior tree

INTRODUCTION

Collaborative construction robots, such as robotic excavators (Jin et al., 2021), offer a promising alternative to relieve construction workers from both physically and cognitively demanding tasks. For safe and productive co-excavation, effective communication between humans and robots is essential to align task goals of the team. Human operators traditionally communicate their task intents through complex joystick control (Jin et al., 2021). The process demands high multi-tasking skills in sensing, planning, and operation, which may cause excessive human workload and safety issues (Lee et al., 2022). In contrast, natural language is an intuitive way for humans to express task intents. Instructing robots through natural language has been widely explored in industrial manufacturing, autonomous driving, and household service (Tellex et al., 2020).

However, robots may struggle to interpret human instructions and generate a reliable execution plan, particularly high-level goal-oriented instructions. Unlike formal robot languages, human languages are inherently abstract and ambiguous. For instance, people prefer issuing high-level goal-oriented instructions such as "Dig a trench over there," rather than step-by-step commands such as "Move forward 1 meter and" and "Low down the bucket 0.5 meter" to specify every action in detail. Many existing studies focused on extracting goals from simple instructions and using classical planners to generate task plans for fulfilling the goals (Pramanick et al., 2020; Tran et al., 2023). They cannot effectively handle high-level instructions that describe long-horizon tasks. Besides, the generated execution plan lacks the flexibility to suit dynamic and unstructured construction environments.

Compared to the rigid task plan, behavior tree (BT) is a control architecture that can adapt to dynamic environments while enabling the translation of human instructions into executable task plans. BTs have been widely employed in many robotic applications such as object manipulation and ground/aerial navigation (Iovino et al., 2022). They provide a fallback mechanism for conditional checks, allowing robots to react dynamically to failures through alternatives (Colledanchise and Ögren, 2018). Besides, the tree-like structure of BT mirrors the hierarchical nature of tasks and associated task instructions. Its modularity and extendibility ease the translation of human language instructions into BT structures.

This paper presents a framework for enabling humans to guide a robotic excavator using both goal-oriented and action-oriented instructions. The framework translates high-level human instructions to low-level robot actions to accomplish specified tasks. The remainder of this paper is organized as follows. The next section introduces the background and related work. The third section outlines the proposed framework with its four modules. The fourth section presents a case illustration to prove the framework's feasibility, and the last section concludes the study with key findings.

BACKGROUND AND RELATED WORK

Using language instructions to guide robotic tasks requires converting language to something readily for robots to grasp and accordingly generating execution plans. Traditional studies modeled robot behaviors using action sequences. The approaches resulted in rigid task plans that are poorly suited to dynamic environments. In contrast, this paper focuses on BT due to their modularity, reusability, flexibility, and adaptability, which make BT well-suited for modeling complex robotic tasks in unstructured construction environments.

Robot Understanding of Human Task Instructions

Humans guide robot tasks using either action-oriented instructions (AOIs) or goal-oriented instructions (GOIs), aligning with the principles of hierarchical task planning. Humans typically model a task in a hierarchical break-down structure, as shown in Figure 1. A task can be either a composite task/subtask that is executed in a sequence of primitive actions, or just one single primitive action. Primitive actions are the atomic elements in the task hierarchy and can be directly executed by robots by running the corresponding low-level routine. Each performed action may impose effects on the world and change the world states such as robot pose and object position. In this way, humans can issue AOIs by specifying detailed actions (e.g., "Go forward 2 meters" and "Pick up the cup") or GOIs by expressing desired outcomes (e.g., "Bring me the cup").

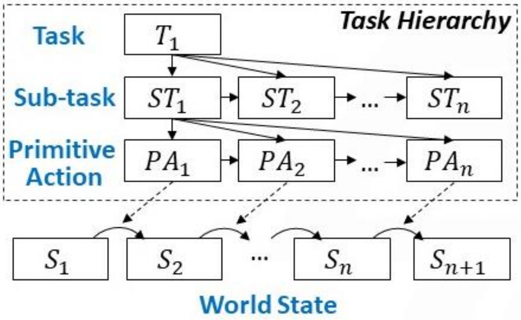


Figure 1: Task hierarchical break-down structure.

To execute instructions effectively, robots need to extract intents—either goals or actions—and generate corresponding task plans. To interpret AOIs, traditional research applied various language parsing techniques, such as CCG parsing (Suddrey et al., 2017) and deep-learning parsing (Sarkar et al., 2023), for extracting actions, binding arguments, and action orders. The extracted actions are matched with the predefined templates in an action library and organized as an action sequence, linking the abstract intent semantics with robot control functions (Lu and Chen, 2017). Differently, understanding GOIs requires parsing embedded goals into logic-based representations that serve as inputs for classical task planners (e.g., PDDL planners (Pramanick et al., 2020; Tran et al., 2023)) to generate task plans. However, existing methods struggle with processing GOIs for long-horizon tasks (e.g. excavation) in two aspects: 1) data limitations: parsing approaches require extensive training data and often fail to handle complex or novel instructions., and 2) computational constraints: classical planners suffer from exponential growth in search space as the number of actions and states increases, leading to long computation times.

Recent studies leveraged LLMs to simultaneously interpret GOIs and plan the interpreted tasks. SayCan framework ranked the admissible atomic behaviors (i.e., action-object pairs) for completing the given home service instruction and grounded the actions to the current scene based on the coupled affordance functions (Ahn et al., 2022). ProgPrompt utilized LLM's strength in code understanding to generate executable plans as programs (Singh et al., 2023). However, the generated plans are all limited to deterministic action sequences, which lack error tolerance and are ill-suited for dynamic environments. Complex tasks managed through long action sequences also require frequent cumbersome replanning when task conditions change.

Behavior Trees for Robot Behavior Modeling

A behavior tree (BT) is a directed rooted tree structure that describes and controls the execution flow of the robot’s behaviors (Colledanchise and Ögren, 2018). Within a BT, *internal nodes* are known as *control flow nodes* to specify the control-flow logic of behaviors, while *leaf nodes* are *execution nodes* defining the robot’s executable behaviors. Nodes are connected by arrows from *parent* to *child*. The *root* is the node without parents while leaf nodes do not have children. Internal nodes have at least one child that could either be a leaf node or internal node itself.

BTs operate by propagating signals called "ticks" from the root node down to its children at regular intervals. If a leaf node receives a tick, it executes and immediately returns a status (*success*, *failure*, or *running*) back up to the root. The tree control flow is determined by this feedback, guiding the robot to proceed to the next leaf node, wait for the current running node, or terminate the tree execution. Table 1 summarizes the types of nodes and their corresponding symbols. Leaf nodes are specified into Action and Condition, while internal nodes include Sequence, Fallback, Parallel, and Decorator types, depending on how they manage child execution.

Table 1. Node types of behavior tree

Node Type	Definition	Symbol
Action	A leaf node that controls the robot to perform actions and returns *success*, *failure*, or *running* based on the execution results.	Text
Condition	A leaf node that checks the world state against the specified condition and returns *success* or *failure* based on the evaluation.	Text
Sequence	An internal node that executes its child nodes in order until one child returns *failure* or all children return *success*.	→
Fallback	An internal node that executes its child nodes in order until one child returns *success* or all children return *failure*.	?
Parallel	An internal node that executes its child nodes in ‘parallel’ until a certain amount of child nodes return *success* or all children return *failure*.	⇉
Decorator	An internal node that modifies a single child through user-defined policy. For example, a *repeat* decorator can execute its child node until the child node returns a specified times of *success* or one time *failure*.	◇

Figure 2 illustrates an example of BT for a trenching task. demonstrating how a robot excavates multiple zones to a specified depth through a work zone loop. This loop includes moving to appropriate positions and executing a digging cycle, which includes sequential actions such as *MoveToDig*, *Dig*, *MoveToDump*, and *Dump* to shape the terrain. During execution, the BT is ticked in a top-down and left-right order until it reaches a terminal state. The figure shows a snapshot of the BT ticking in which the tree is waiting for the running *MoveToDump* node to complete. If the node succeeds, the next action *Dump* executes; if it fails, the tree terminates the task. This example highlights the advantages of BTs over traditional action sequences in modelling robot behaviors:

- **Modularity and reusability**: BTs can be generated and extended with reusable nodes and subtrees. Their modular nature enhances the readability and manageability of complex tasks.
- **Flexibility and adaptability**: BTs provide the fallback mechanism for conditional checks, allowing robots to react dynamically to failures through alternatives.

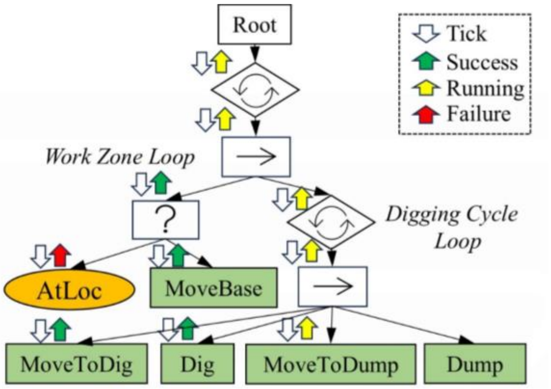


Figure 2: Behavior tree graph of trenching task performed by excavators.

Early research work translated AOIs into BTs using CCG parsing and analytical mapping the parsing elements to BT nodes (Suddrey et al., 2022). But the methods could only deal with a limited set of simple instructions. Some recent studies employed LLMs to generate BTs in an end-to-end manner from human instructions (Izzo et al., 2024). They prepared a large synthetic dataset of instruction-tree pairs and fine-tuned lightweight LLMs for learning the BT generation. However, the generated BTs were not grounded with robot skills or environments, limiting their real-world applicability. Other studies proposed multi-step pipelines for converting GOIs to executable BTs (Chen et al., 2024; Zhou et al., 2024), leveraging LLMs to decompose high-level goals into subgoals and generate BTs through analytical expansion or optimal expansion. The pipelines have showed promising results in terms of understanding accuracy and success rates of task execution.

FRAMEWORK OF HUMAN LANGUAGE-INSTRUCTED ROBOTIC EXCAVATION

This paper presents a framework to enable humans to guide a robotic excavator using both goal-oriented and action-oriented instructions, as shown in Figure 3. The objective of the framework is to translate goal-oriented/action-oriented instructions into BTs as reliable execution plans. The proposed framework consists of four modules: Interpretation and reasoning, knowledge management, structure analysis and parsing, and BT generation. Each module's functionality and enabling approaches are described below.

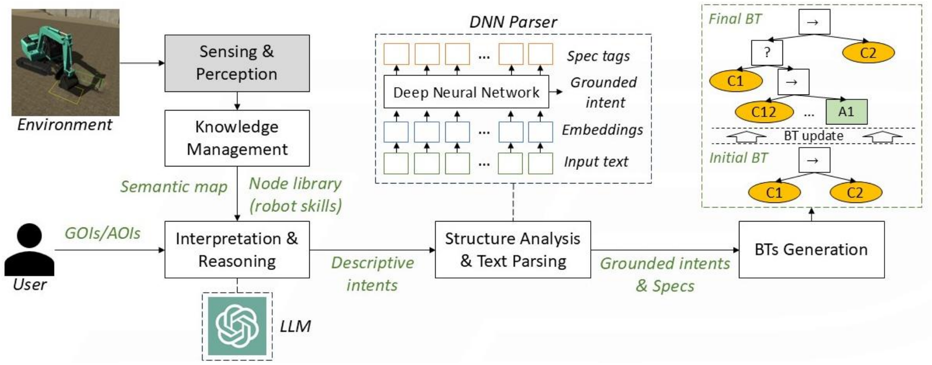


Figure 3: Framework of human language-instructed robotic excavation.

Interpretation and Reasoning. This module aims at breaking down human-issued GOIs/AOIs into smaller intents as high-level language plans. To ensure that these decomposed intents are viable for execution, the knowledge of robot capable skills and current working environment is given as input for the decomposition. LLM serves as the reasoning engine for the decomposing process, utilizing its versatility in natural language understanding, commonsense reasoning, and text generation. Prompt-based learning techniques are adopted to guide the generation of relevant outputs while maintaining a structured format for better downstream processing.

Knowledge Management. This module organizes and maintains critical information about the environment and the robot's skills for LLM-based reasoning. The environmental information may include 1) elevation maps of target trench and current terrain, 2) mask maps of dig and dump sites, and 3) occupancy maps of obstacles (Terenzi and Hutter, 2023). Semantic maps are used to deliver map information for high-level task planning while a database stores the map grid values for low-level planning and control. Besides, the robot's skills are documented within a node library that defines the robot's executable actions and conditions. Relevant information includes action names, action arguments, preconditions and postconditions of the action, and action descriptions.

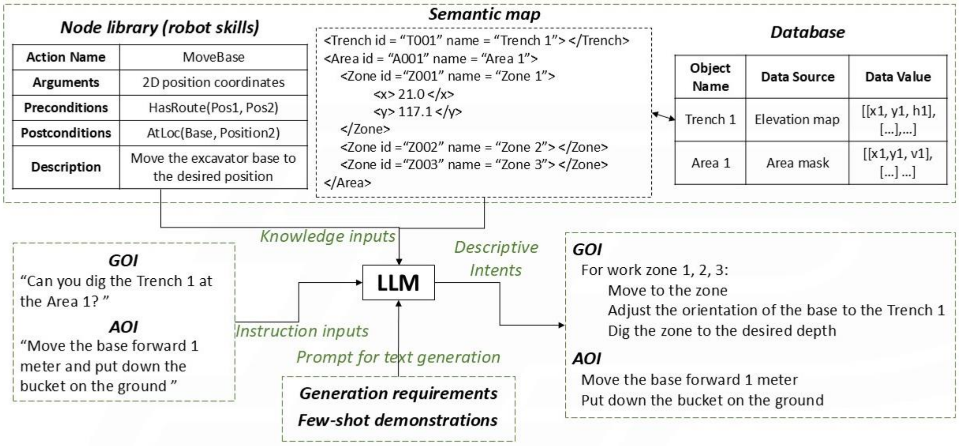
Structure Analysis and Parsing. This module processes descriptive intents to extract grounded intents and relevant task specifications. The structure analysis is first conducted to identify task loops and their positions within the high-level plan. Then the parsing algorithm extracts grounded intents along with associated parameters. A deep learning parser such as BERT-based parser can be developed to accurately extract the intents.

BT Generation. This module creates an executable BT based on the extracted intents and specifications. An initial tree is constructed by mapping the intents to the predefined action/post-condition node from the node library and sequencing the nodes according to the task logic. Then the tree is iteratively pre-executed for finding failed condition nodes. The failed nodes are replaced or updated with corresponding enabled action nodes to ensure smooth execution. BT updating algorithms such as back-chaining can be used in this process.

CASE ILLUSTRATION

This section presents a case for illustrating the feasibility of the proposed framework. In this case, human workers issue both goal-oriented instructions (GOIs) and action-oriented instructions (AOIs) to guide a robotic trenching task. The human worker and the robot are assumed to share all the necessary task information. To initialize the task, the human worker may issue a GOI such as “Can you dig the Trench 1 at the Area 1?” If any task failure clues are detected during supervising, the human worker can proactively intervene by giving an AOI to correct robot actions, such as “Move the base forward 1 meter and put down the bucket on the ground”. The following paragraphs present how these two instruction examples are translated into BTs.

Interpretation and Reasoning Using Context Knowledge. Figure 4 demonstrates the conversion of the GOI and AOI to descriptive intents. The GOI is interpreted into a high-level language plan with a loop structure to reflect work zone loop, while the AOI is broken down into a simple list of human-specified actions. To improve the effectiveness of the LLM-based generation, the prompts provided to the LLM include both generation requirements and few-shot demonstrations. For instance, “Please generate the descriptive intents based on the robot actions predefined in the node library” is a requirement instructing the LLM to align the outputs to the robot capabilities.



Node library (robot skills)

Action Name	MoveBase
Arguments	2D position coordinates
Preconditions	HasRoute(Pos1, Pos2)
Postconditions	AtLoc(Base, Position2)
Description	Move the excavator base to the desired position

Semantic map

```
<Trench id = “T001” name = “Trench 1”> </Trench>
<Area id = “A001” name = “Area 1”>
   <Zone id =“Z001” name = “Zone 1”>
         <x> 21.0 </x>
         <y> 117.1 </y>
   </Zone>
   <Zone id =“Z002” name = “Zone 2”> </Zone>
   <Zone id =“Z003” name = “Zone 3”> </Zone>
</Area>
```

Database

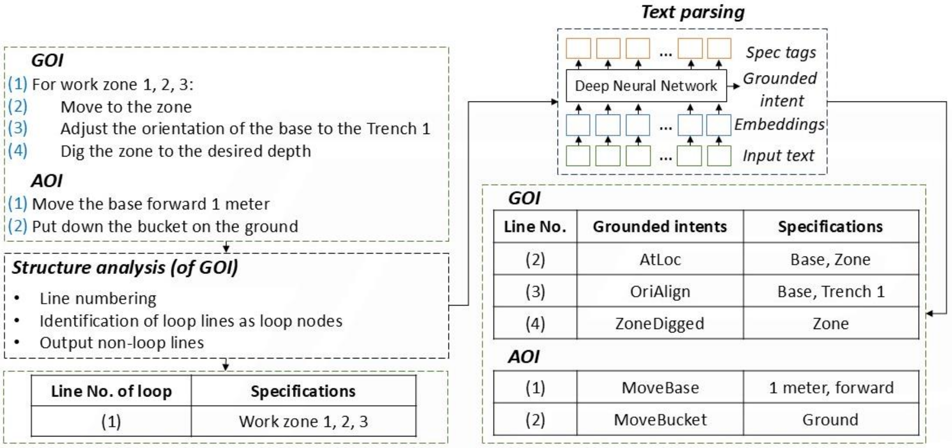
Object Name	Data Source	Data Value
Trench 1	Elevation map	[[x1, y1, h1], [...],...]
Area 1	Area mask	[[x1,y1, v1], [...] ...]

GOI
“Can you dig the Trench 1 at the Area 1? ”
AOI
“Move the base forward 1 meter and put down the bucket on the ground ”

GOI
For work zone 1, 2, 3:
    Move to the zone
    Adjust the orientation of the base to the Trench 1
    Dig the zone to the desired depth
AOI
Move the base forward 1 meter
Put down the bucket on the ground

Figure 4: Illustration of converting GOI and AOI to decomposed descriptive intents based on given knowledge.

Extraction of Grounded Intents and Associated Specifications. Figure 5 shows the extraction of grounded intents and associated specifications from descriptive intents. The GOI is analyzed with its structure to identify the loop and its position. The remaining non-loop descriptions undergo text parsing to obtain grounded intents and specifications. To facilitate training and matching with robot skills, the extracted grounded intents are annotated using the action and condition node names from the predefined node library.


Text parsing
Spec tags
Deep Neural Network
Grounded intent
Embeddings
Input text
GOI
(1) For work zone 1, 2, 3:
(2) Move to the zone
(3) Adjust the orientation of the base to the Trench 1
(4) Dig the zone to the desired depth
AOI
(1) Move the base forward 1 meter
(2) Put down the bucket on the ground
Structure analysis (of GOI)
• Line numbering
• Identification of loop lines as loop nodes
• Output non-loop lines

Line No. of loop	Specifications
(1)	Work zone 1, 2, 3

GOI

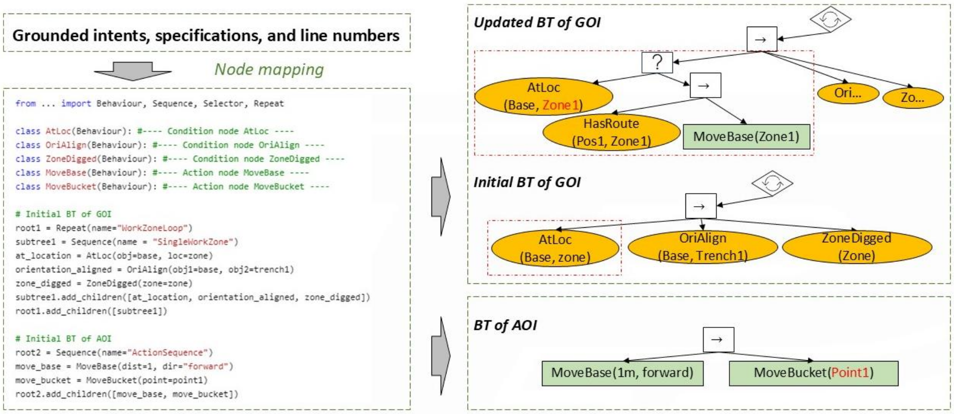
Line No.	Grounded intents	Specifications
(2)	AtLoc	Base, Zone
(3)	OriAlign	Base, Trench 1
(4)	ZoneDigged	Zone

AOI

Line No.	Grounded intents	Specifications
(1)	MoveBase	1 meter, forward
(2)	MoveBucket	Ground

Figure 5: Illustration of extracting grounded intents and associated specifications.

Generation of BTs through Tree Expanding. Figure 6 illustrates the process of BT generation given the grounded intents, specifications, and line numbers. The grounded intents can be mapped with predefined action nodes and condition nodes through keyword matching. Specifications are used to populate the arguments of node functions. Line numbers help determine the layouts and sequencing of nodes in the BT structure. The left part of Figure 6 demonstrates coding examples of initial BT trees using the py_trees library. Once the initial BT of GOI is built and executed, failed post-condition nodes are updated with enabled action nodes to achieve the post-conditions. The right part of Figure 6 shows the updating example. The failed condition node *AtLoc* is replaced by a subtree, which contains the action node *MoveBase* and its pre-condition node *HasRoute*. The example also highlights cases where invalid node arguments (marked in red) are identified and corrected with valid ones, ensuring the correctness of the updated BT. Figure 7 demonstrates the final resulting BTs of the two instructions after updating. It is worth noting that an action node is not limited to representing a primitive action but can also be a subtree that addresses a complex goal, leveraging the reusability of BTs. For instance, the post-condition *ZoneDigged* is addressed by a loop subtree of sequential actions in the GOI case.



```
from ... import Behaviour, Sequence, Selector, Repeat

class AtLoc(Behaviour): #---- Condition node AtLoc ----
class OriAlign(Behaviour): #---- Condition node OriAlign ----
class ZoneDigged(Behaviour): #---- Condition node ZoneDigged ----
class MoveBase(Behaviour): #---- Action node MoveBase ----
class MoveBucket(Behaviour): #---- Action node MoveBucket ----

# Initial BT of GOI
root1 = Repeat(name="WorkZoneLoop")
subtree1 = Sequence(name = "SingleWorkZone")
at_location = AtLoc(obj=base, loc=zone)
orientation_aligned = OriAlign(obj1=base, obj2=trench1)
zone_digged = ZoneDigged(zone=zone)
subtree1.add_children([at_location, orientation_aligned, zone_digged])
root1.add_children([subtree1])

# Initial BT of AOI
root2 = Sequence(name="ActionSequence")
move_base = MoveBase(dist=1, dir="forward")
move_bucket = MoveBucket(point=point1)
root2.add_children([move_base, move_bucket])
```

Figure 6: Illustration of generating BTs through tree expanding.

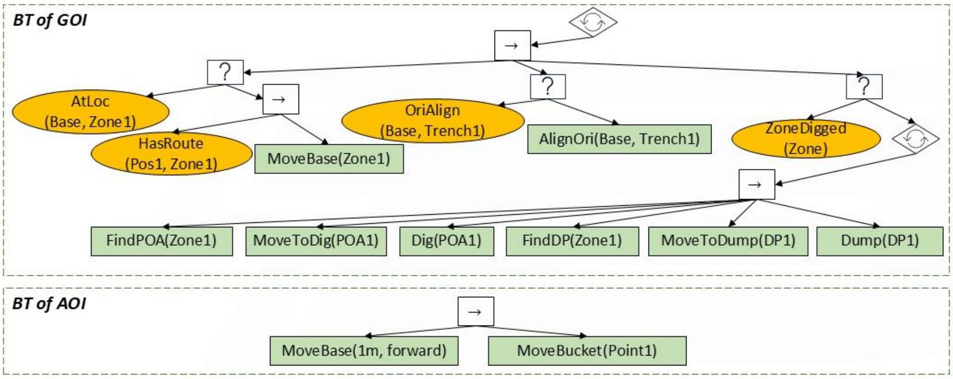


Figure 7: Resulting BTs of two instructions.

CONCLUSION

This paper proposed a framework for enabling humans to guide a robotic excavator using both goal-oriented and action-oriented instructions. The proposed framework is demonstrated through a case illustration of human workers instructing a trenching task using the two instruction types. The case highlights the expressiveness of BTs in representing long-horizon complex tasks like excavation in a human-readable format. It also demonstrates the potential of BTs in facilitating the translation of human language instructions into reliable adaptive task plans. Future studies are needed to address the following key challenges: 1) optimizing prompt design for efficient and accurate instruction decomposition, 2) effective detecting and updating invalid node arguments during BT generation, and 3) automating the pipeline for real-world robot task applications.

REFERENCES

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R.J., Jeffrey, K., Jesmonth, S., Joshi, N.J., Julian, R., Kalashnikov, D., Kuang, Y., Lee, K.-H., Levine, S., Lu, Y., Luu, L., Parada, C., Pastor, P., Quiambao, J., Rao, K., Rettinghouse, J., Reyes, D., Sermanet, P., Sievers, N., Tan, C., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Xu, S., Yan, M., Zeng, A., 2022. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances.

Chen, X., Cai, Y., Mao, Y., Li, M., Yang, Z., Shanghua, W., Yang, W., Xu, W., Wang, J., 2024. Efficient Behavior Tree Planning with Commonsense Pruning and Heuristic.

Colledanchise, M., Ögren, P., 2018. Behavior trees in robotics and AI: An introduction. CRC Press.

Iovino, M., Scukins, E., Styrud, J., Ögren, P., Smith, C., 2022. A survey of Behavior Trees in robotics and AI. Rob Auton Syst 154, 104096. https://doi.org/10.1016/j.robot.2022.104096

Izzo, R.A., Bardaro, G., Matteucci, M., 2024. BTGenBot: Behavior Tree Generation for Robotic Tasks with Lightweight LLMs.

Jin, Z., Pagilla, P.R., Maske, H., Chowdhary, G., 2021. Task Learning, Intent Prediction, and Adaptive Blended Shared Control with Application to Excavators. IEEE Transactions on Control Systems Technology 29, 18–28. https://doi.org/10.1109/TCST.2019.2959536

Lee, J.S., Ham, Y., Park, H., Kim, J., 2022. Challenges, tasks, and opportunities in teleoperation of excavator toward human-in-the-loop construction automation. Autom Constr 135, 104119.

Lu, D., Chen, X., 2017. Interpreting and extracting open knowledge for human-robot interaction. IEEE/CAA Journal of Automatica Sinica 4, 686–696. https://doi.org/10.1109/JAS.2017.7510628

Pramanick, P., Barua, H.B., Sarkar, C., 2020. DeComplex: Task planning from complex natural instructions by a collocating robot. IEEE International Conference on Intelligent Robots and Systems 6894–6901. https://doi.org/10.1109/IROS45743.2020.9341289

Sarkar, C., Mitra, A., Pramanick, P., Nayak, T., 2023. tagE: Enabling an Embodied Agent to Understand Human Instructions. Findings of the Association for Computational Linguistics: EMNLP 2023 8846–8857. https://doi.org/10.18653/v1/2023.findings-emnlp.593

Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., Garg, A., 2023. ProgPrompt: Generating Situated Robot Task Plans using Large Language Models, in: Proceedings - IEEE International Conference on Robotics and Automation. Institute of Electrical and Electronics Engineers Inc., pp. 11523–11530. https://doi.org/10.1109/ICRA48891.2023.10161317

Suddrey, G., Lehnert, C., Eich, M., Maire, F., Roberts, J., 2017. Teaching Robots Generalizable Hierarchical Tasks Through Natural Language Instruction. IEEE Robot Autom Lett 2, 201–208. https://doi.org/10.1109/LRA.2016.2588584

Suddrey, G., Talbot, B., Maire, F., 2022. Learning and Executing Re-Usable Behaviour Trees From Natural Language Instruction. IEEE Robot Autom Lett 7, 10643–10650. https://doi.org/10.1109/LRA.2022.3194681

Tellex, S., Gopalan, N., Kress-Gazit, H., Matuszek, C., 2020. Robots That Use Language. Annu Rev Control Robot Auton Syst 3, 25–55. https://doi.org/10.1146/annurev-control-101119-071628

Terenzi, L., Hutter, M., 2023. Towards Autonomous Excavation Planning 1–32.

Tran, D., Li, H., He, H., 2023. AI Planning from Natural-Language Instructions for Trustworthy Human-Robot Communication, in: International Conference on Social Robotics. Springer, pp. 254–265.

Zhou, H., Lin, Y., Yan, L., Zhu, J., Min, H., 2024. LLM-BT: Performing Robotic Adaptive Tasks based on Large Language Models and Behavior Trees. https://doi.org/10.1109/ICRA57147.2024.10610183