# Learning Autonomous Humanoid Loco-manipulation Through Foundation Models

Jin Wang, Rui Dai, Weijie Wang

Humanoids and Human Centered Mechatronics (HHCM)
Istituto Italiano di Tecnologia Italy
wang.jin@iit.it

**Abstract:** Humanoid robots with behavioral autonomy have consistently been regarded as ideal collaborators in our daily lives and promising representations of embodied intelligence. Compared to fixed-based robotic arms, humanoid robots offer a larger operational space while significantly increasing the difficulty of control and planning. Despite the rapid progress towards general-purpose humanoid robots, most studies remain focused on locomotion ability with few investigations into whole-body coordination and tasks planning, thus limiting the potential to demonstrate long-horizon tasks involving both mobility and manipulation under open-ended verbal instructions. In this work, we propose a novel framework that learns, selects, and plans behaviors based on tasks in different scenarios. We combine reinforcement learning (RL) with whole-body optimization to generate robot motions and store them into a motion library. We further leverage the planning and reasoning features of the large language model (LLM), constructing a hierarchical task graph that comprises a series of motion primitives to bridge lower-level execution with higher-level planning. Experiments in simulation and real-world using the CENTAURO robot show that the language model based planner can efficiently adapt to new loco-manipulation tasks, demonstrating high autonomy from free-text commands in unstructured scenes.

**Keywords:** LLM, loco-manipulation, humanoid robot

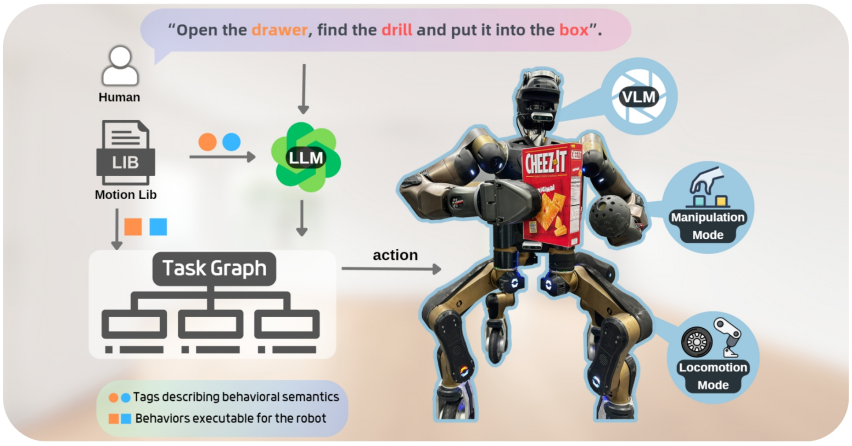


Figure 1: Humanoid robot CENTAURO picks objects with the planning of the *task graphs* generated by the LLM. The '*Motion lib*' consists of various action and sensing behaviors with 'tags' describing the semantic content of different behaviors. The VLM selects '*Manipulation Mode*' and '*Locomotion Mode*' based on different task scenarios.

CoRL 2024 Workshop on Whole-Body Control and Bimanual Manipulation (CoRL 2024 WCBM).

# 1 Introduction

Maintaining autonomy during the execution of a task in a real-world environment is both essential and challenging for robots, especially when performing tasks that require interaction with surroundings and manipulation of objects. This involves robots being able to reason the given semantic instructions and plan their behaviors while using multi-modality to perceive and infer affordances and spatial geometric constraints of the environment to determine proper motions.

Recently, vibrant advances in robotics learning have made it a promising avenue for manipulation and locomotion [1, 2, 3, 4, 5]. Learning-based methods, such as reinforcement learning (RL), have become effective tools for task-oriented action generation [6, 7, 8] while facilitating generalization across diverse scenarios. However, extending learning algorithms to humanoid robots remains a challenge stemming from the exponential increase in training costs induced by high degrees of freedom (DoF) and the difficulty of deployment on real robots under dynamic constraints. Meanwhile, the rise of large language models (LLMs) and their remarkable capabilities in robotic planning [9, 10] have made it possible to perform logical reasoning and construct hierarchical action sequences for complex tasks. By integrating observations from different modalities, those models can be used for extracting features of objects and environments for robot perception and decision-making. Nevertheless, limitations to the utilization of LLM in humanoid robots exist, particularly in complex whole-body motion control and precise coordination between body parts.

To address these issues, we first recognized that directly outputting whole-body trajectories for a real-world multi-joint system through simulation training is inefficient and impractical. Therefore, we adopt a decomposed training strategy that modularly selects the actuation components related to given tasks, and project lower-dimensional space trajectory on the whole-body space with a unified motion generator. The trained actions are stored as skill units in the motion library. We utilize the LLM's ability to decompose complex semantic instructions consisting of multiple sub-tasks and design a modular user interface as model's input. The LLM selects skills from the motion library and arranges a sequence of actions, referred to as task graphs. Furthermore, the 3D features extracted from captured 2D images and depth data can be integrated with the visual language model (VLM) and robotic intrinsic characteristics, acting as a robotic motion morphology selector.

In this study, we present a language model based framework enabling task reasoning and autonomous behavior planning towards humanoid loco-manipulation. We use a decomposed training strategy that modularly selects the components needed for specific tasks and maps low-dimensional space trajectories to the whole-body space with a unified motion generator. The trained actions are stored as skill primitive in a motion library. We adopt the LLM to decompose complex instructions consisting of multiple sub-tasks that select skills from the motion library and arranges a sequence of actions, referred to as a task graph. By leveraging the interaction of distilled spatial geometry and 2D observation with a visual language model (VLM) to ground knowledge into a motion morphology selector to choose appropriate actions in single- or dual-arm, legged or wheeled locomotion. We further illustrate through experiments how our framework can be learned and deployed on a high-DoF, hybrid wheeled-leg robot, performing zero-shot online planning under human instruction.

Our summary of the main contributions of this work includes:

- We present a novel method for autonomous behavior planning in loco-manipulation tasks through grounded language model that adapts to humanoids, quadrupeds, and mobile manipulators, extending the scope of LLMs in domains of embodied intelligence.
- Enabling robots to autonomously reason and select motion morphology in single- or dual-arm, legged, or wheeled locomotion during tasks by leveraging the interaction between distilled spatial geometry and 2D observations through VLMs.
- We adopt layered policy for learning humanoid motions, which involves task-specific training and synthesizing motion primitive through whole-body optimization, demonstrating that generated actions can robustly and seamlessly perform different tasks.

Experiments in simulation and real-world show the proposed framework can efficiently adapt to new tasks, demonstrating high autonomy of humanoid loco-manipulation from open-set instructions.

## 2 Methodology

### 2.1 Autonomous Behavior Planning Framework

We illustrate how the proposed framework enables the humanoid robot CENTAURO [11] to autonomously perform loco-manipulation guided by semantic instructions. As shown in Fig. 2, we divide the pipeline into four main interrelated sectors that are learned and deployed sim-to-real manner. The motion generation sector selects RL training configurations for specific tasks and conducts training in parallel. The trajectory obtained from the training is provided as a reference to the optimizer, which ultimately generates whole-body motion skills and the skills will be stored in the motion library. The user input sector contains a user interface as well as pre-defined basic prompts, function options, and motion library, all of which together constitute the textual material fed to the LLM. After receiving a command, the task planning sector generates a hierarchical task graph using the LLM. Once the task graph is loaded, it is interpreted as a Behavior Tree to guide the robot's execution. When a task requires selecting the motion morphology, depth-sensing information is invoked and distilled into 2D images and geometric features. These data, along with the task state and prompts, are fed to the VLM, which then selects the morphology capable of achieving the goal. Through the coordination of these sectors, the study facilitates semantic command understanding and zero-shot behavioral planning and action execution for CENTAURO robot.

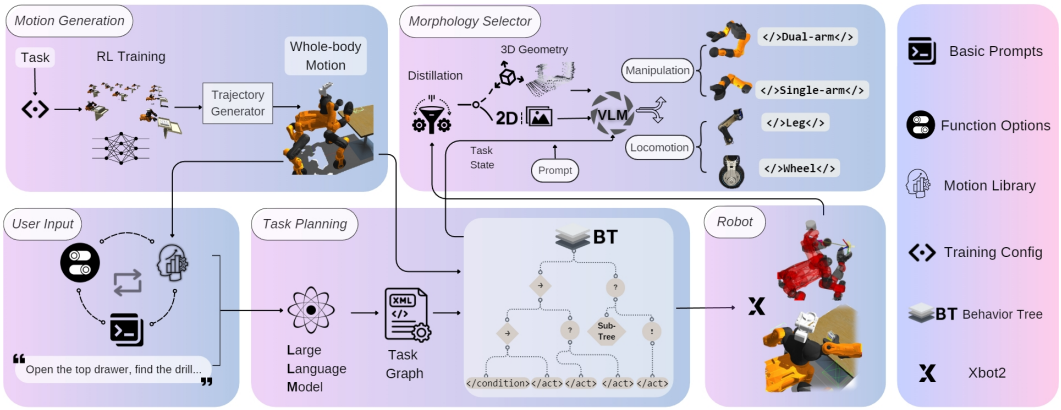


Figure 2: Overview of the Framework. **Motion generation** is assigned for learning and training whole-body motion skills for new tasks and storing them in the motion library. **User input** includes received task instructions and initialization prompt sets. **Task planning** generates a task graph that guides the robot's behavior and passes action commands to the real robot. **Morphology Selector** is used for motion mode determination in specific sub-tasks, selecting the appropriate morphology for locomotion and manipulation based on spatial affordances and robot intrinsic features.

### 2.2 Learning Whole-body Motion Generation

In this section, we present the learning process for generating whole-body motion based on various tasks. To reduce the action space, we separate the robot's upper body from its legs, allowing the floating base to be heuristically limited, ensuring feasibility and avoiding unfeasible leg motions. For single-arm tasks, the action space consists of the right arm's 6-DoF joint angles, 6-DoF floating base movements, torso yaw, and gripper joint angle, while the left arm remains fixed. For dual-arm tasks, the left arm's 6-DoF joint angles are added. Observations include the states of the corresponding targets and the robot's upper body joint states. Using PPO for efficiency, the RL output is an upper body joint position trajectory, and the skill policies are trained using a reward formulation that balances reaching, rotation, grasping, task completion, and smoothness. The reduced robot space from RL is mapped to the whole-body joint space by solving an optimal control problem to merge

the upper body trajectory with the whole-body dynamics, ensuring the resulting motion is feasible. This is achieved using a trajectory generator that optimizes for both upper body reference trajectories and whole-body dynamic feasibility for task completion.

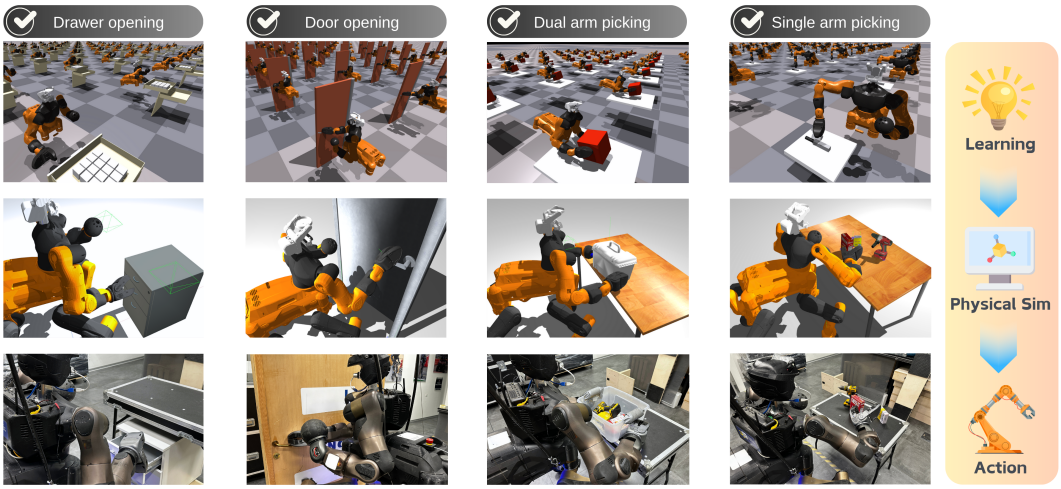


Figure 3: Whole-body tasks learning illustration in training, simulation and real-world settings.

After learning task-orient motion skills, we constructed a motion library to host these primitives, which consists of attribute and functional descriptions of these actions, and the corresponding learning-based whole-body policies. Then, the LLM can reason the attribute-function descriptions to create sequences of actions to be executed based on different tasks and generate a task graph to invoke the execution of each node without additional training or demonstration.

### 2.3 Humanoid Robot Task Planning with grounded language models

Migrating foundation models from a fixed robotic arm to a humanoid robot with a floating base presents numerous issues and challenges. The addition of robotic components not only imposes complex dynamic constraints, making it difficult to coordinate and control various parts. It also requires addressing the potential for different manipulation modes inherent in human-like structures, as well as the increased DoF for spatial mobility by the addition of wheels and legs. Due to the construction of our motion library, the usage of the LLM for planning no longer requires additional considerations for constraints such as self-collision or self-posture balance maintenance. This allows more focus on the decomposition of given tasks, and the selection of the robot's morphology.

Humans utilize common sense and learned experiences to extract the affordances of objects they manipulate and select appropriate movement based on the estimation of geometric constraints of the environment. Inspired by this, we leverage VLM to implement similar functionalities in robots. First, we include descriptions of the robot's structure and functions, and the robot's achievable range of motion in the prompts $\mathbf{p}_V$. While determining the morphology for a manipulation task $T_m$, the robot utilizes 2D and depth images from its head camera. Object detection and pose estimation algorithms [12, 13] are invoked to acquire the position and orientation of the target object $\mathbf{v}_c \in \mathbb{R}^6$, which is then transformed into the robot's coordinate system $\mathbf{v}_R \in \mathbb{R}^6$. The VLM $\mathcal{V}$, based on the current task state $\mathbf{s}$, the scene's 2D images $\mathbf{I}^h_{\text{scene}}$, and the target object's 6D pose $\mathbf{v}_R$, generates the robot's manipulation morphology $\mathbf{x}_m$ for the task scenario. For locomotion tasks, the robot uses the depth information from its pelvis depth camera to generate the point cloud $\mathbf{P}_c$, which is down-sampled to create a voxel grid $\mathbf{V}_g$. This spatial information containing the current moving path together with the 2D image $\mathbf{I}^p_{\text{scene}}$ and the task state are finally fused to select the robot's locomotion morphology $\mathbf{x}_l$ using the VLM $\mathcal{V}$.

$$\mathbf{x}_m = \mathcal{V}(\mathbf{s}, \mathbf{I}^h_{\text{scene}}, \mathbf{v}_R, \mathbf{p}_V) \tag{1}$$

$$\mathbf{x}_l = \mathcal{V}(\mathbf{s}, \mathbf{I}^p_{\text{scene}^p}, \mathbf{V}_g, \mathbf{p}_V) \tag{2}$$

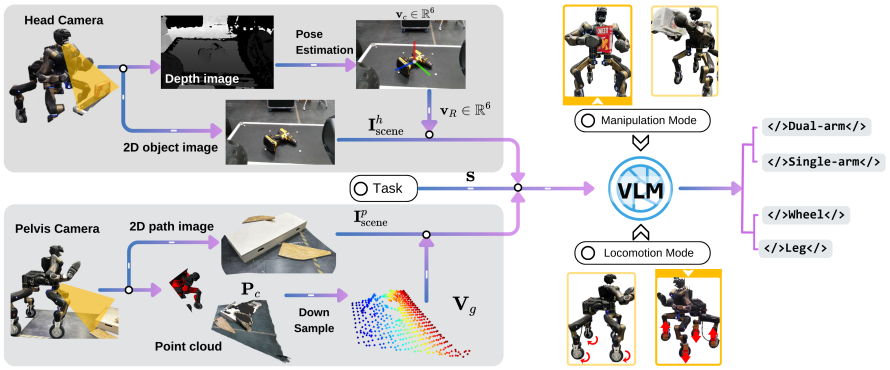


Figure 4: **Robotic Morphology Selector** extracts spatial geometric data and 2D observations from the physical environment upon receiving the language-conditioned task state and interacts with the VLM incorporating the grounded robot's affordances, so as to provide the optimal motion morphology that meets the requirements of given task scenario during manipulation and locomotion process.

To obtain the desired response from LLM, it is necessary to impose constraints on the input. In the user interface, we define three types of constraints. The basic prompt provides a description of task background and characteristics of the robot, as well as interpretation of the user command and the output format. The motion library offers a catalog of learned skills and their description. Function option module offers specifications of the added functions developed for humanoid robot and determines whether these predefined functions are invoked during planning. Such as, if the morphology selection is chosen, the LLM will incorporate the morphology selector based on the task scenario; otherwise, this function will not be considered (See the Appendix). This approach allows for systematic construction of prompts and modular addition of constraints, thereby enhancing the flexibility of planning. We utilize BehaviorTree (BT) [14] as an intermediate bridge to convert high-level instructions into executable low-level skill sequences. BT provides a hierarchical structure for guiding actions and making decisions for the robot, which is composed of nodes with different effects. With a pre-defined motion library, LLM can generate a task graph consisting of learned motions and BT nodes, build it in an XML file, which constructs the complete BT. Thus realizing the robot's behavior planning with LLM by giving verbal instruction.

### 2.4 Failure Detection and Recovery

In order to determine whether a task is successfully completed or deviates during execution, we try to incorporate a failure detection and recovery mechanism into the task graph. In our work, to take advantage of the visual language model's capability of understanding and reasoning about images, we utilize visual questions and answers (VQA) as perceptual behaviors to determine the current state of the robot performing the task, such as in the task of 'picking the box' by giving the robot's camera image and asking "Is the box being held?", the VLM will respond to the query by answering "`Yes`" or "`No`". Proprioceptive sensing like torque and distance has also been developed as behaviors to detect the potential failures in specific tasks, like during the tasks requiring grasping, detecting the torque on the gripper can be a reference of whether the object is being held.

During the execution of the behavior tree, the node will return three signals: $success$, $failure$, and $running$, and the behavior tree will guide the execution of the behavior according to the returned signals. After the action node is executed, the condition node can be added to decide whether the current task is successful or returns the current state of the target object. For instance, in the process of grasping and lifting an object, after the completion of grasping, a condition node `IsObjectHeld` can be added to decide whether the object has been successfully grasped or not. In this scenario, the node will activate the `Grip force` and `Visual Q&A` perception behavior, which will obtain the torque of the gripper and ask the VLM "Is the (Target Object) held by the gripper". Only if there is a torque on the gripper and the VLM answers "Yes", then the node will return a success signal. The

behavior tree will continue to execute the subsequent nodes. If it returns a failure, the recovery node is activated and the robot will try to grasp the object again.

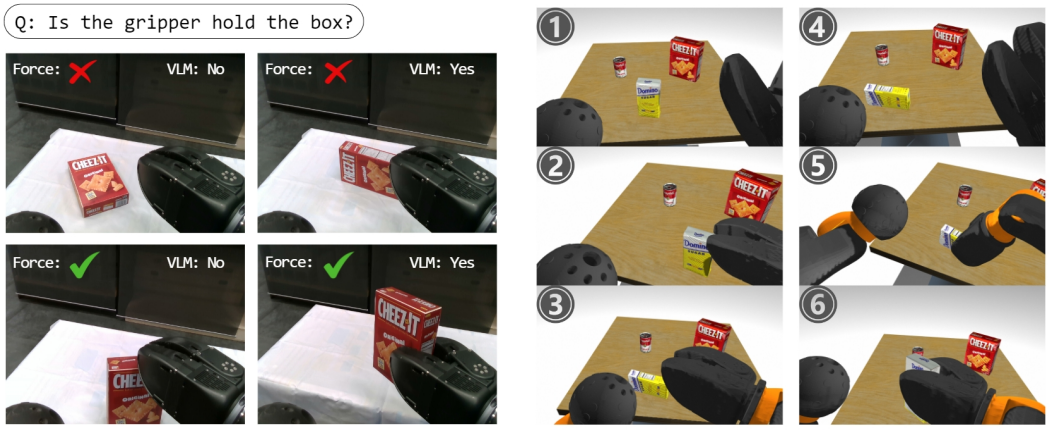


Figure 5: (a) Failure detection using a combination of perception modals. By asking VLM, the `visual Q&A` behavior can reason the state of the task, while using the torque sensor, the `Grip force` will return the torque on the gripper. (b) Graphs 1, 2, and 3 show the robot's first attempt to pick up an object. After the perception behaviors detected that the gripper did not successfully grasp the object in graph 3, then the robot tried again and successfully picked the object as shown in image 4, 5, 6.

## 3 Experiment

We demonstrate the experiments in both simulation and real-world environment using objects that can be commonly found in daily life. Our robot is a centaur-like humanoid robot, supported by four legs with wheels. The robot has two arms with one claw gripper on its right arm. There are two depth cameras, one is on the head and another is in the pelvis position. We use Xbot to achieve real-time communication between the underlying actuators and the control commands. We use Isaac Gym [15] as a training environment and validate the planned whole-body motion using Gazebo [16]. We access the `gpt-4o` model as the LLM planner and the `gpt-4v` model as the VLM from OpenAI API [17].

Table 1: Comparison of different methods towards various robotic tasks. TLE(time limit exceeded)

| Task | WB-MPC | | LLM-Planner | | LLM-Planner+MS | |
|---|---|---|---|---|---|---|
| | Succ ↑ | Avg. Time ↓ | Succ ↑ | Avg. Time ↓ | Succ ↑ | Avg. Time ↓ |
| Move to target | 84% | 25.7s ± 12.3 | 96% | 36.8s ± 10.6 | 92% | 40.1s ± 29.3 |
| Approach Object | 72% | 24.3s ± 10.6 | 88% | 34.9s ± 8.6 | 96% | 39.8s ± 22.1 |
| Open door | 64% | 45.6s ± 13.5 | 84% | 35.2s ± 12.6 | 88% | 42.5s ± 18.6 |
| Pick object | 68% | 38.9s ± 24.2 | 80% | 38.6s ± 9.6 | 84% | 44.3s ± 22.3 |
| Pick and place object | 60% | 50.2s ± 47.3 | 72% | 49.2s ± 23.6 | 84% | 54.7s ± 32.1 |
| Open drawer and pick object | 0% | TLE | 64% | 105.4s ± 27.6 | 72% | 126.3s ± 33.6 |

**Loco-manipulation Tasks Planning** We experimentally validate the efficacy of our method in behavior learning, planning, and decision-making by implementing the framework and evaluating its performance on a wheel-legged humanoid robot executing long-horizon tasks in response to semantic commands. we selected six loco-manipulation tasks and compared our method's performance

with traditional whole-body MPC control methods [18] and conducted 25 experiments for each tasks separately, the simulation results are shown in Table 1.

The experimental results indicate that using an LLM to plan pre-trained motion primitives for executing loco-manipulation tasks yields a higher success rate compared to WB-MPC, particularly for complex tasks such as 'pick and place object' and tasks requiring long-term planning such as 'open drawer and pick object.' However, for some simpler tasks, the time required by the LLM-Planner is slightly longer than that of WB-MPC, attributable to the time taken for online API requests and BT construction. Adding Morphology Selector (MS) into the task planning process results in a modest increase in the average planning time, but it concurrently enhances the success rate of tasks, especially those involving interaction with the surrounding environment and objects.

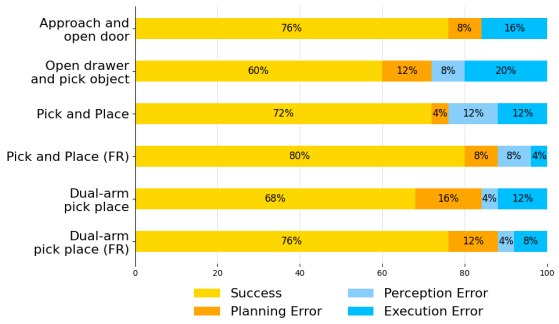


Figure 6: Average rate of CENTAURO robot successfully performing various LLM planning tasks, and failure caused by different type of errors during the tasks.

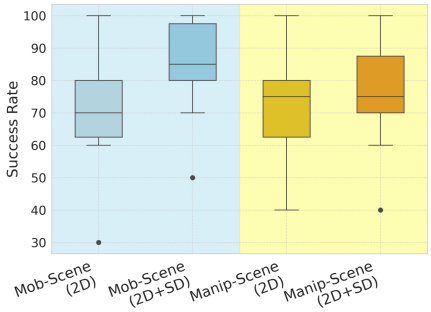


Figure 7: Success rate of the morphology selector for different scenarios. "2D" and "SD" are image and Spatial Data inputs.

**Autonomous Behavior Planning and Failure Detection** We validate the ability of LLM to plan motion primitives for different loco-manipulation tasks. Experiments were conducted on tasks requiring a combination of perceptions and actions. We recorded the success rate and the impact of different errors of 4 representative tasks and provided quantitative evaluations in Fig.6. The results show the LLM based planner can effectively plan for semantic instructions based on learned skills and guide the robot to complete a variety of tasks according to the action sequences, achieving a desired success rate ($\geq 60\%$) on real-world robot. And adding failure detection and recovery (FR) to the planning increases the success rate of task execution. Whereas selecting multiple functional modules as input also increases the difficulty of planning, and execution errors mainly stem from intricate dynamical constraints on the actions and misalignment of the floating sensing with robot execution.

**Morphology Selection Towards Different Scenarios** We investigated whether a VLM can zero-shot determine robot's morphology based on task scenarios. We picked ten scenarios each for manipulation and locomotion in both simulation and real-world environments. We compared the success rate of the VLM's morphology selection using 2D image input only versus image combined with spatial geometric as input. Each scenario was tested 10 times under both inputs, as shown in Fig. 7. We found the morphology selector effectively chooses the optimal mode for everyday object manipulation and mobile environment with a high average success rate. Compared to solely image input, adding spatial information improves the selector's accuracy, particularly in determining locomotion modes and adapting to complex scenarios (paths with obstacles of varying types and heights), thus leveraging the robot's affordances and leading to robust execution.

**Long-horizon Task** A long-horizon task refers to a task that needs to be completed over an extended time period, typically involving the execution of multiple subtasks and decision-making across several steps. In this study, we define long-horizon tasks based on the number of robotic actions included in the task. When the task described by the given instructions comprises four or more robotic actions, we classify it as a long-horizon task. Here, robotic actions refer to those included in the action library. For example, the task 'Pick the box' consists of three actions: $< ObjectDetect >$,

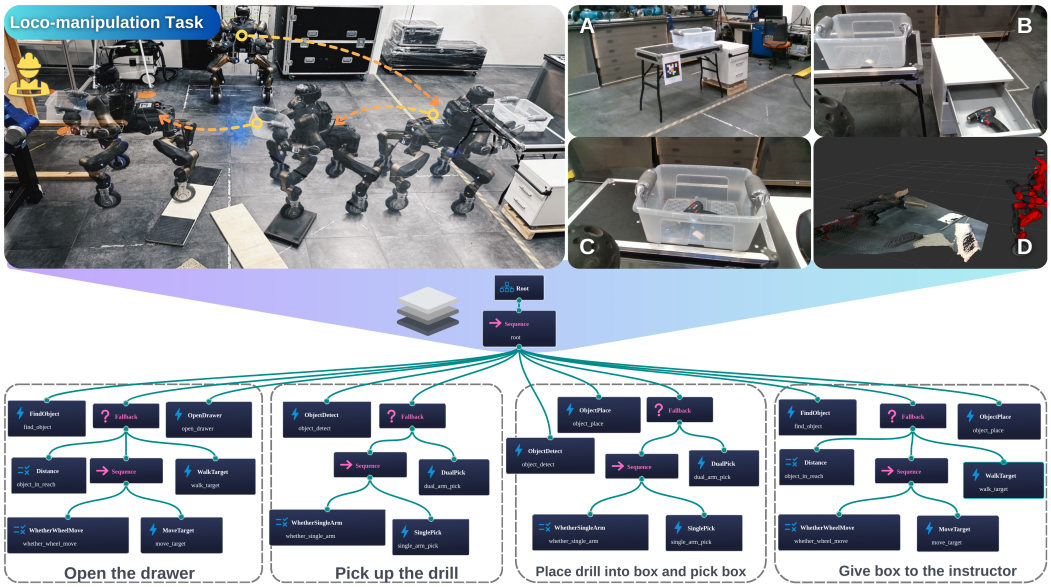


Figure 8: **Overall look of the long-horizon task** Images above show the timelapse of roll outs to robot motion trajectory. A. semantic navigation using AprilTag. B. object detection and pose estimation. C. manipulation morphology selection. D. locomotion morphology selection. The BehaviorTree below shows the details of LLM task planning.

$< MoveTarget >$, and $< DualPick >$, and is thus categorized as a simple motion task. In contrast, the long-horizon task described in Fig. 8 requires the robot to perform multiple actions, including $< FindObject >$, $< MoveTarget >$, $< ObjectDetect >$, $< SinglePick >$, and $< OpenDrawer >$ et., with a longer time span required for task completion.

Qualitative results including time-lapse shots of robot motion execution and a Behavior Tree mapped out by LLM are shown in Fig.8. We demonstrate that our framework can synthesize sequences of motion primitives based on designed user input and accurately infer the logic of semantic knowledge while selecting robotic morphology of locomotion and manipulation according to the environment and state of the task. We found that language-based behavior planner exhibits greater versatility and adaptability to more complex tasks compared to existing methods.

## 4 Conclusion

In this work, we present a framework that enables humanoid robots to learn, select, and plan behaviors, integrating knowledge and robotic affordance to perform embodied tasks. We evaluate the framework's efficiency and versatility through real-world experiments and long-horizon tasks. Despite achieving expected results, there are **limitations**: the motion library's size restricts the range of task commands, and learning of new skills requires separate training optimization, hindering generalization from existing actions. Moreover, the system struggles to handle external disturbances and collisions, lacks real-time linguistic interaction during the task and has limited capability for re-planning in response to unexpected tasks.

Future work will focus on enriching the robot's behavior lib, as well as improving the prompts system, so that the LLM can better plan and optimize behavioral sequences automatically based on the robot's intrinsic mobility, manipulation, and perceptual strengths, thus enabling to perform more complex mobile manipulation tasks. Another direction is to improve the dynamic planning and multiconditional reasoning capability of the framework. This includes behavioral replanning in response to external perturbations or the introduction of artificial subtasks during a task.

**Acknowledgments**

If a paper is accepted, the final camera-ready version will (and probably should) include acknowledgments. All acknowledgments go at the end of the paper, including thanks to reviewers who gave useful comments, to colleagues who contributed to the ideas, and to funding agencies and corporate sponsors that provided financial support.

## References


[1] O. Kroemer, S. Niekum, and G. Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of Machine Learning Research*, 22(30): 1–82, 2021. URL http://jmlr.org/papers/v22/19-804.html.

[2] H. Zhang, G. Solak, G. J. G. Lahr, and A. Ajoudani. Srl-vic: A variable stiffness-based safe reinforcement learning for contact-rich robotic tasks. *IEEE Robotics and Automation Letters*, 9(6):5631–5638, 2024. doi:10.1109/LRA.2024.3396368.

[3] Y. Ma, F. Farshidian, T. Miki, J. Lee, and M. Hutter. Combining learning-based locomotion policy with model-based manipulation for legged mobile manipulators. *IEEE Robotics and Automation Letters*, 7(2):2377–2384, 2022. doi:10.1109/LRA.2022.3143567.

[4] Q. Zhang, P. Cui, D. Yan, J. Sun, Y. Duan, A. Zhang, and R. Xu. Whole-body humanoid robot locomotion with human reference, 2024.

[5] S. Li, J. Wang, R. Dai, W. Ma, W. Y. Ng, Y. Hu, and Z. Li. Robonurse-vla: Robotic scrub nurse system based on vision-language-action model, 2024. URL https://arxiv.org/abs/2409.19590.

[6] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot parkour learning, 2023.

[7] D. Hoeller, N. Rudin, D. Sako, and M. Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88):eadi7566, 2024. doi:10.1126/scirobotics.adi7566. URL https://www.science.org/doi/abs/10.1126/scirobotics.adi7566.

[8] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme parkour with legged robots, 2023.

[9] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control, 2023.

[10] A. Z. Ren, A. Dixit, A. Bodrova, S. Singh, S. Tu, N. Brown, P. Xu, L. Takayama, F. Xia, J. Varley, Z. Xu, D. Sadigh, A. Zeng, and A. Majumdar. Robots that ask for help: Uncertainty alignment for large language model planners. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=4ZK8ODNyFXx.

[11] J. Wang, A. Laurenzi, and N. Tsagarakis. Autonomous behavior planning for humanoid loco-manipulation through grounded language model, 2024. URL https://arxiv.org/abs/2408.08282.

[12] B. Wen, W. Yang, J. Kautz, and S. Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects, 2024.

[13] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. In *Conference on Robot Learning (CoRL)*, 2018. URL https://arxiv.org/abs/1809.10790.

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

[15] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021.

[16] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, 2004. doi:10.1109/IROS.2004.1389727.

[17] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[18] I. Dadiotis, A. Laurenzi, and N. Tsagarakis. Whole-body mpc for highly redundant legged manipulators: Experimental evaluation with a 37 dof dual-arm quadruped. In *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, pages 1–8, 2023. doi: 10.1109/Humanoids57100.2023.10375215.

# Appendix

## Contents

## A1 Robot System Setup

### A1.1 Robot hardware

Our robot is a centaur-like robot platform. The upper body of the robot is humanoid in design and is similar in size to the average human to adapt to both dual-arm and single-arm manipulation. The robot's mobility relies on its quadrupedal lower body and maintains whole-body balance to cope with a variety of terrain conditions and perform loco-manipulation tasks. Moreover, to improve the robot's mobility on flat ground, wheel modules are integrated underneath each leg and can control the direction and steering of the wheels.

The robot's whole body consists of 38 actuatable joints. The robot's torso is mounted on the pelvis of the lower body via yaw joints, allowing the upper body to rotate in the transverse plane. Each arm of the robot includes 6 DoF, where the right hand gripper contains one extra DoF that controls its opening and closing. The robot's legs are designed to provide an omni-directional wheeled motion and articulated legged locomotion, with each leg containing six degrees of freedom, allowing for positioning, orientation, and rotation of the wheeled-leg module.

The perception system of the robot consists of two on-board RealSense Depth Camera D435i, one located in the robot's head and the other in the robot's pelvis, which are used to provide 2D images and depth information of the surrounding environment and objects. The complete computing system consists of two on-board computing units (ZOTAC-EN1070K PC, COM Express conga-TS170) for system communication and real-time robot control and an external pilot PC (Inter Core i9-13900HX CPU @3.90GHz, NVIDIA GeForce RTX 4090) for task planning and sensory data processing as well as a user interface.

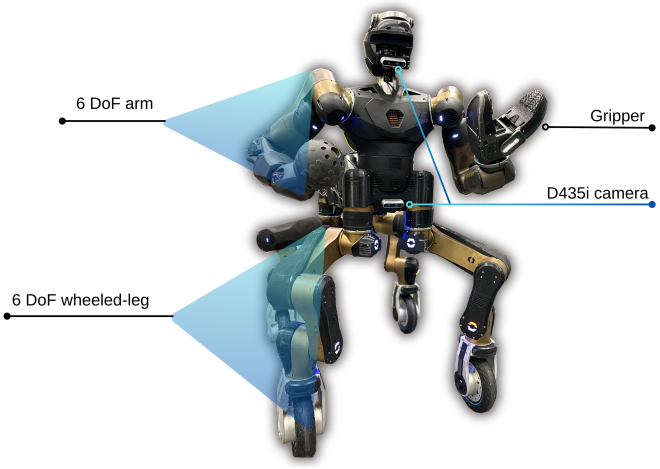


Figure A1: Robot hardware setup

### A1.2 Robot software

We use XBotCore, a cross-platform, real-time, open-source software designed for interfacing with low-level hardware components of robots [1]. This innovative tool enables effortless programming and management of various robotic systems by offering a standardized interface that conceals the intricacies of the hardware. Additionally, a proprietary CartesI/O motion controller [2] handles higher-order motion instructions. It is capable of managing multiple responsibilities and restrictions, prioritized according to the demands of specific situations. Through solving a series of quadratic programming (QP) challenges, each linked to a unique priority tier, the controller ensures optimal performance across all preceding priority stages.

## A2 Details of Robot Learning

We utilize Proximal Policy Optimization (PPO) [3] for training our tasks, employing a multi-layer perceptron within an actor-critic framework. The network architecture for the drawer opening, door opening, and dual-arm picking tasks consists of layers with [256, 128, 64] units while the picking task uses layers with [256, 128, 64] units. The activation function applied across all tasks is ELU. Below, we detail the observations, task-specific rewards ($r_{task}$), and reward parameters for each task.

### A2.1 drawer opening

First, we define the frame of the drawer handle. The x-axis of the handle points towards the robot, while the z-axis points upwards. The handle's inward direction is aligned negatively along the x-axis, and the upward direction is consistent with the z-axis. The task reward is defined as

$$r_{task} = \alpha_7 r_{around} + l_{drawer} * r_{around} + l_{drawer} \tag{A1}$$

where $r_{around} = 0.5$ when the gripper's top link is above the handle's position and the bottom link is below the handle's position, otherwise $r_{around} = 0$. $l_{drawer}$ represents the length by which the drawer has been pulled.

The observations and reward parameters for this task are listed in Tab. 1 and 2.

| |
|---|
| normalized upper body joints position |
| upper body joints velocity * 0.1 |
| drawer pulled length |
| vector from gripper to drawer handle |

Table 1: observations of drawer opening task

| | |
|---|---|
| $\alpha_1$ | 2.0 |
| $\alpha_2$ | 0.0 |
| $\alpha_3$ | 0.5 |
| $\alpha_4$ | 7.5 |
| $\alpha_5$ | 7.5 |
| $\alpha_6$ | 0.01 |
| $\alpha_7$ | 0.7 |
| $\beta$ | 0.04 |

Table 2: reward parameters of drawer opening task

### A2.2 door opening

The door handle has the same frame as the drawer handle. The task reward is defined as

$$r_{task} = \alpha_7 r_{around} + angle_{handle} * r_{around} + angle_{handle} + angle_{door} \tag{A2}$$

where $r_{around}$ is the same setting as the drawer opening task and $angle_{handle}$ represents the angle by which the door handle has been pushed. $angle_{door}$ is the angle of the opened door.

The observations and reward parameters for this task are listed in Tab. 3 and 4.

| |
|---|
| base pose |
| right arm joints position |
| door handle pose |
| gripper pose |
| door handle angle |
| door opened angle |

Table 3: observations

| | |
|---|---|
| $\alpha_1$ | 2.0 |
| $\alpha_2$ | 0.0 |
| $\alpha_3$ | 1.5 |
| $\alpha_4$ | 7.5 |
| $\alpha_5$ | 2.0 |
| $\alpha_6$ | 0.01 |
| $\alpha_7$ | 0.125 |
| $\beta$ | 0.02 |

Table 4: parameters

### A2.3 single arm picking

We define the object's upward direction as aligning negatively along the x-axis, and the inward direction as aligning negatively along the z-axis. This orientation encourages the gripper to adopt a top-to-bottom pose, facilitating a proper grasp of the object. The task reward is defined as

$$r_{task} = \alpha_7 r_{around} + h \tag{A3}$$

where $r_{around}$ is the same setting as the previous tasks with the corresponding object frame and $h = 1$ if the object is been picked up, otherwise $h = 0$.

The observations and reward parameters for this task are listed in Tab. 5 and 6.

| |
|---|
| base pose |
| right arm joints position |
| object pose |
| gripper pose |

Table 5: observations

| | |
|---|---|
| $\alpha_1$ | 7.5 |
| $\alpha_2$ | 0.0 |
| $\alpha_3$ | 5.0 |
| $\alpha_4$ | 2.5 |
| $\alpha_5$ | 7.5 |
| $\alpha_6$ | 0.01 |
| $\alpha_7$ | 0.7 |
| $\beta$ | 0.1 |

Table 6: parameters

### A2.4 dual arm picking

In the dual arm picking task, the distance $d_l$ and $d_r$ represents the left end-effector and right end-effector to the left and right side of the object, respectively. The task reward is defined as

$$r_{task} = h \tag{A4}$$

where $h = 1$ if the object is been picked up, otherwise $h = 0$.

The observations and reward parameters for this task are listed in Tab. 7 and 8.

| |
|---|
| base pose |
| two arms joints position |
| object pose |
| left end-effector pose |
| right end-effector pose |
| vector from object left side to left end-effector |
| vector from object right side to right end-effector |

Table 7: observations

| | |
|---|---|
| $\alpha_1$ | 2.0 |
| $\alpha_2$ | 2.0 |
| $\alpha_3$ | 0.0 |
| $\alpha_4$ | 0.0 |
| $\alpha_5$ | 7.5 |
| $\alpha_6$ | 0.01 |
| $\alpha_7$ | 0.0 |
| $\beta$ | 0.0 |

Table 8: parameters

## A3 Details of Whole-body Optimization

The trajectory optimization problem essentially constitutes a Nonlinear Programming (NLP) challenge characterized by a predetermined quantity of nodes and intervals. Its canonical formulation typically adheres to Eq.(A5)

$$\begin{cases} \min_{\mathbf{x}(.),\mathbf{u}(.)} \int_0^T L(\mathbf{x}(t),\mathbf{u}(t),t)dt \\ \text{s.t. } \dot{\mathbf{x}}(t) = \boldsymbol{f}(\mathbf{x}(t),\mathbf{u}(t),t) \\ \boldsymbol{g}_1(\mathbf{x}(t),\mathbf{u}(t),t) = 0 \\ \boldsymbol{g}_2(\mathbf{x}(t),\mathbf{u}(t),t) \leq 0 \end{cases} \tag{A5}$$

the standard formulation necessitates conversion into a discrete programming format . Subsequently, we discrete the state and input variable as the follow sets, $N$ is the node number

$$\mathcal{X} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix}; \mathcal{U} = \begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_N \end{bmatrix} \tag{A6}$$

then the general optimization form Eq.(A5) becomes Eq.(A7)

$$\begin{aligned} J &= \sum_{i=0}^{N} L_i(\mathbf{x}_i, \mathbf{u}_i) \\ \dot{\mathbf{x}}_i &= \boldsymbol{f}(\mathbf{x}_i, \mathbf{u}_i) \, , i = 0, \cdots N \\ \mathbf{C}_{\min} \leq \mathbf{C}(\mathbf{x}_i, \mathbf{u}_i) &\leq \mathbf{C}_{\max}, i = 0, \cdots N \end{aligned} \tag{A7}$$

where , $\mathbf{C}(\mathbf{x}_i, \mathbf{u}_i)$ is the discrete form of equality and inequality constrain, $\mathbf{C}_{\min}$ is the lower limit, $\mathbf{C}_{\max}$ is the upper limit. Specifically, in order to keep the trajectory feasible, we should shape the constrains as:

$$\begin{aligned} \mathbf{q}^0 &= \mathbf{q}_{\text{init}} && \text{initial position} \\ \mathbf{v}^0 &= 0 && \text{initial velocity} \\ \mathbf{q}^k_{\min} \leq \mathbf{q}^k &\leq \mathbf{q}^k_{\max} && \text{position bounds } \forall k \in [1, N-1] \\ \mathbf{v}^k &\leq \mathbf{v}^k_{\max} && \text{velocity bounds} \quad \forall k \in [1, N-1] \\ \dot{\mathbf{v}}^k_{\min} &\leq \dot{\mathbf{v}}^k \leq \dot{\mathbf{v}}^k_{\max} && \text{acceleration bounds} \quad \forall k \in [0, N-1] \\ \mathbf{f}^{z,k}_{c,i} \cdot \mathbf{n}_i > 0, \left\| (\mathbf{f}^{x,k}_{c,i}, \mathbf{f}^{y,k}_{c,i}) \right\|_2 &\leq \mu_i \left( \mathbf{f}^{z,k}_{c,i} \cdot \mathbf{n}_i \right) && \text{leg contact force bounds} \quad \forall k \in [0, N-1] \end{aligned} \tag{A8}$$

where $\mathbf{f}_{c,i} = [\mathbf{f}^x_{c,i}, \mathbf{f}^y_{c,i}, \mathbf{f}^z_{c,i}]$ is the $i$-th leg contact force. At the end of programming, its function of the whole body trajectory is to realize the motion learned from RL framework, we implement the cost as :

$$L_i(\mathbf{x}_i, \mathbf{u}_i) = \|\mathbf{q}^u_i - \mathbf{q}^*_i\|^2 + \|\mathbf{u}\|^2 \tag{A9}$$

the term $\|\mathbf{q}^u_i - \mathbf{q}^*_i\|^2$ is for merging the gap between RL trajectory and actually feasible trajectroy, $\mathbf{q}^u_i$ is the upper body trajectory from RL, $\mathbf{q}^*_i$ is the upper body trajectory from whole body optimization, $\|\mathbf{u}\|^2$ for reduce the energy of the whole motion.

## A4 Motion Library

We constructed a motion library to house the learned whole-body skills as well as the action and condition nodes used to construct the task graph. The motion library includes information about the skills fed to the LLM, as well as the control code corresponding to each skill. The following Fig. A2, A3 shows the action skills and nodes inside the motion library that LLM can choose to invoke to construct the task graph.

```
### Action Node ###

<HomingPose>: 'name'='homing_pose'; 'type'=general; 'label'=start the robot to a initial position;
'description'=control the robot to power up and back to the initial robot pose.

<FindObject>: 'name'='find_object'; 'type'=general; 'label'=look around for object;
'description'=control the robot to turn on the head camera, and rotates itself to find 'object' and
acquire its 3D position.

<MoveTarget>: 'name'='move_target'; 'type'=wheel; 'label'=approach to target with wheels;
'description'=control the robot to approach to the target location using wheel motion (require knowing
3D position of 'target').

<WalkTarget>: 'name'='walk_target' ;'type'=leg; 'label'=approach to target with legs;
'description'=control the robot to approach to the target location using leg motion (require knowing
3D position of 'target').

<ObjectDetect>: 'name'='object_detect'; 'type'=general; 'label'=object detection and pose estimation;
'description'=using the head camera to detect and estimate the position and pose of the
'target_object'.

<ObjectPlace>: 'name'='object_place'; 'type'=general; 'label'=place object to a target position;
'description'=control the robot to put the object to a target position. (require knowing 'target' 3D
position).

<OpenDoor>: 'name'='door_open'; 'type'=general; 'label'=open the door; 'description'=control the robot
to open the door. (require knowing 3D position of 'door').

<SinglePick>: 'name'='single_arm_pick'; 'type'=single_arm; 'label'=grasp object and pick it up;
'description'=control the robot to grasp the target object with right arm, and pick it up (require
knowing 'target_object' position and pose).

<DualPick>: 'name'='dual_arm_pick'; 'type'=dual_arm; 'label'=hold object with dual arms and pick it
up. 'description'=control the robot to hold the target object with dual arms, and pick it up (require
knowing 'target_object' position and pose).

<OpenDrawer>: 'name'='open_drawer'; 'type'=general; 'label'=open the drawer. 'description'=control the
robot to open the drawer. (require knowing 3D position of 'drawer').
```

Figure A2: Action nodes in the motion library, where the blue nodes are based on learned whole-body motion skills.

```
### Condition Node ###

<Distance>: 'name'='object_in_reach'; 'type'=general; 'label'=is object in reach;
'description'=measure the distance from the object to the robot. if it is larger than 80cm then return
to fail. (require knowing 'object' 3D position)

<WhetherSingleArm>: 'name'='whether_single_arm'; 'type'=general; 'label'=select robotic morphology
based on manipulation task; 'description'=apply VLM to reason whether to use a single arm or dual arm
to manipulate object, can be used to make decisions before picking actions.

<WhetherWheelMove>: 'name'='whether_wheel_move'; 'type'=general; 'label'=select robotic morphology
based on locomotion task; 'description'=apply VLM to reason whether to use wheel or leg to move, can
be used to make decisions before locomotion actions.

<IsActionSuccess>: 'name'='is_action_completed': 'type'=general; 'label'=reason about the success of
the action; 'description'=apply VLM to reason whether the previous manipulation action is successful,
if not, repeat the action one time in behavior tree.
```

Figure A3: Condition nodes with different functions in the motion library.

## A5 Motion Morphology Selection

In this section, we show the task scenarios used for the motion morphology selection experiments.

### A5.1 Manipulation Scenarios

For the robot manipulation morphology selection experiments included six simulated and four real-world scenarios. We conducted ten morphology selections for each scenario, and before each trial, the positions and poses of the objects in the scenarios were reset. We applied the same prompts for all manipulation morphology selections, with the instructions for each scenario shown in Fig. A4.

### A5.2 Locomotion Scenarios

The robot locomotion morphology selection experiments included six simulated and four real-world scenarios, as shown in Fig.A5. We conducted ten morphology selections for each scenario, and before each trial, the positions of the robot and obstacles in the scenarios were reset. We applied the same prompts for all locomotion morphology selections.

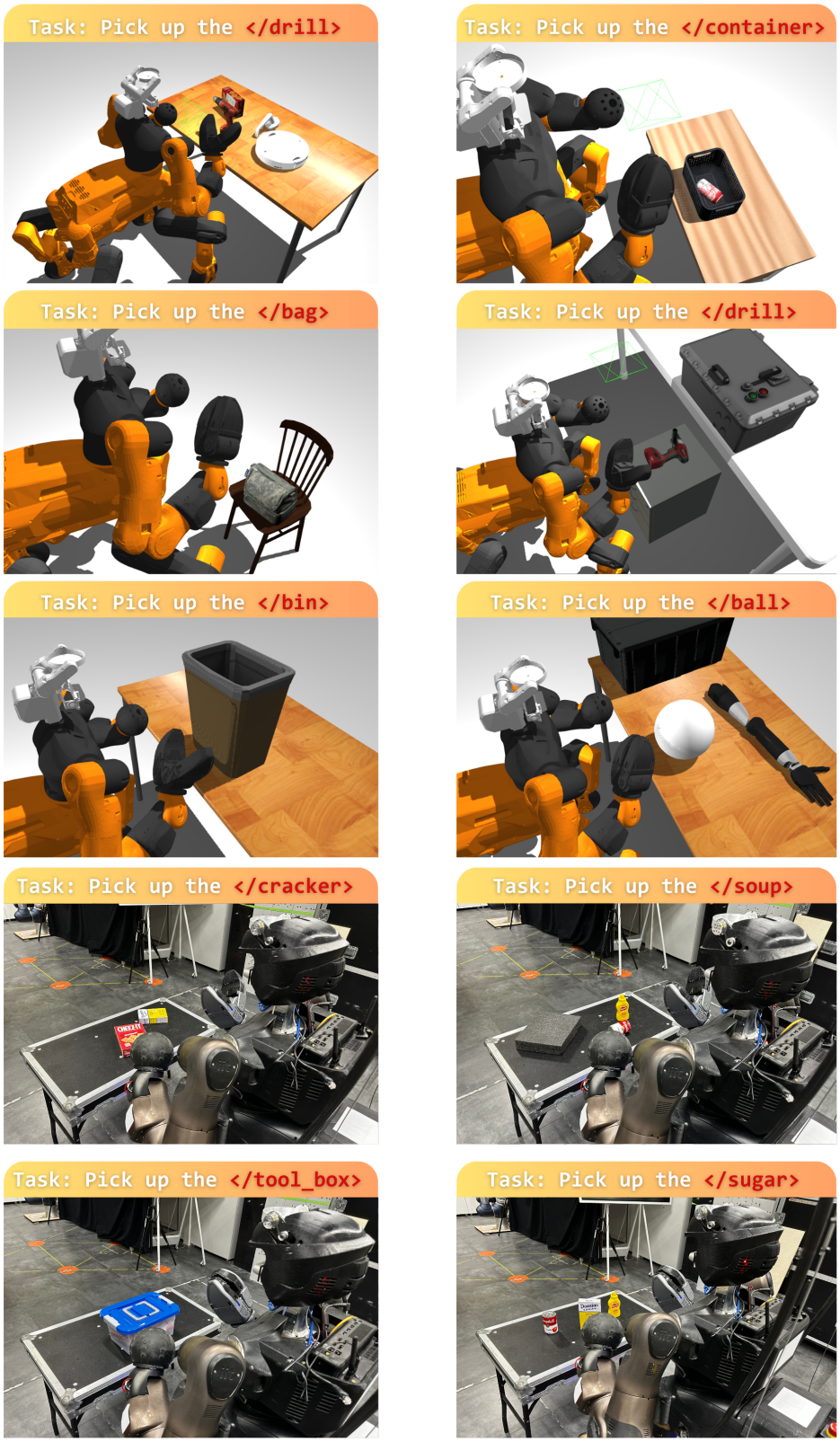


Figure A4: Task scenarios for manipulation morphology selection experiments.

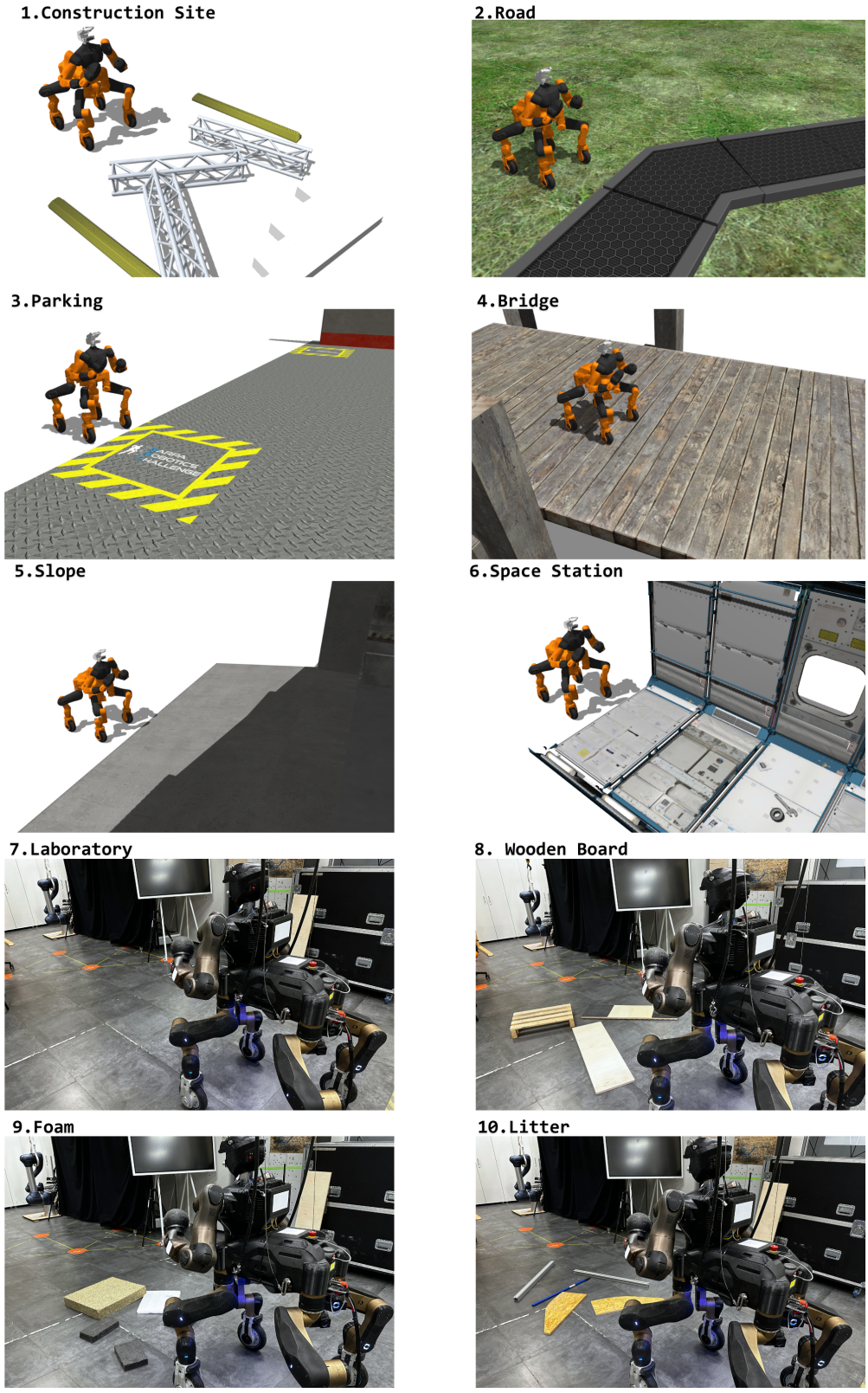


Figure A5: Task scenarios for locomotion morphology selection experiments.

### A5.3 VLM Prompts

The prompt words used for the motion morphology selector are shown in the figures, where the prompt words for manipulation morphology selector will be fed into the VLM along with the received textual task instructions from the Behavior Tree.

The motion morphology selector are packaged as one of the functions in 'User Input' module and it turned 'off' by default. When it needs to be invoked in task planning, it must be enable in 'Function Options' or specified to be set to 'on' when inputting the task instructions.

```
"Suppose you are a humanoid robot and you have two arms, the right arm has a claw gripper
as the end-effector."
"You have two ways to manipulate object: single arm manipulation and dual arm
manipulation."
"You have a camera on your head that can see the object and the environment."
"And you can choose the manipulation method based on the object and task instruction
reasoning."
"When you are doing single arm manipulation, your claw jaw gripper can open up to 10cm."
"You only have the ability to use the jaws to pick up object, regardless of other skillful
grasping methods"
"meaning that you cannot use single arm grasping if the size of the object exceeds the size
of the jaw'opening and closing,"
"or if the object current pose is not capable for grasping with one hand."
"For example: if the given task is 'pick up the drill', and the image shows that a drill is
on the desk."
"Then you will choose to use single arm manipulation, because the size of the drill in the
image can be manipulated with single arm"
"Now you receive the image from the camera and the task below, please answer whether to use
'single arm' or 'dual arm' to do the manipulation."
"(Please answer with only 'single' or 'dual')"
```

Figure A6: Prompts used for Manipulation Morphology Selection.

```
"Suppose you are a robot and you have two ways to move: legs and wheels. "
"You have a depth camera that can obtained the 2D image and point cloud of
the road in front you."

"Now you have to pass the road in front you and here is the 2D image of the
road, and the down sampled point cloud "

"Please choose whether the road should be passed with legs or wheels."
"(Note that wheels are used when the road ahead is flat, or a slope, or the
maximum height of the obstacles is lower than 5 cm."
"And legs should be used when there are obstacles or wooden planks 'maximum
height higher than 5 cm'. )"
"Determine which type of movement the robot should use to pass through the
roadway. "
"(Please answer with only 'leg' or 'wheel', and the data below is the point
cloud)"
```

Figure A7: Prompts used for Locomotion Morphology Selection.

## A6 User Input

The 'User Input' is the module that links the instructor to the language model and contains predefined prompts for initializing the language system environment and limiting the model output, as well as an interface for accepting task commands sent from the user side.

### A6.1 Basic Prompts

Basic prompts provide a description of the task context and robot characteristics, as well as an explanation of user commands and output formatting requirements. As shown below:

```
###Basic Prompts###
"You are now a robot controller, please output a XML file for
constructing a behavior tree to control the robot under the
requirements and given task."
"The robot you control is a centaur like robot, with a humanoid
upper body and four legs, each leg has a wheel at the bottom."
"The robot has two arms, with a claw gripper on the right arm.
It can manipulate objects with two ways: single-arm manipulation
and dual-arm manipulation."
"The robot has two modes of movement: wheel motion and leg motion.
The robot default manipulation and locomotion modes are
'single arm' and 'wheel'."
"The robot has two depth cameras: one located on the head to view
objects, and one on the waist to view the road and terrain ahead."
```

### A6.2 Function Options

We designed a number of functions for the robot and packaged them into condition nodes for selective invocation by the LLM during the planning of the task. These functions include: 'Manipulation Morphology Selector', 'Locomotion Morphology Selector', 'Failure Detection and Recovery'. We add the descriptions of these functions acting as 'Function Options' inside the 'User Input', and set all functions to 'off' state by default. When the instructor expects a function to be added during a task planning, it can be manually set to 'on' or include a declaration to use the function in the instruction.

```
###Function Options###
"The robot has the following functions, all of which are 'off' by
    default."
"When a function is 'on', it need to be involved in planning for the
    given task, and when it is 'off', it should not be used."
"Functions: "

"1. 'manipulation_mode_selector': this function allows the robot to
    add the condition node <WhetherSingleArm> to the planning of
    BehaviorTree, which is used to determine whether the current
    manipulation task should use the 'single_arm' or 'dual_arm' type
    of action."

"2. 'locomotion_mode_selector': this function allows the robot to add
    the condition node <WhetherWheelMove> to the planning of the
    behavior tree, which is used to determine whether the current
    locomotion task should use the 'wheel' or 'leg' type of action."

"3. 'detection_recovery': this allows the robot to add the condition
    node <IsActionSuccess>, which is used to determine whether the
    previous action has been successfully completed and, if not, to
    employ a recovery mechanism that repeat the action."
```

### A6.3 User Interface

The user interface is responsible for accepting task commands from the instructor and combining them with pre-defined prompt for input to the LLM. The complete user input is as follows.

`User Interface`: `hy-motion.github.io/prompt/user_input.ini`

`Motion Library`: `hy-motion.github.io/prompt/motion_library.ini`

`Basic Prompts`: `hy-motion.github.io/prompt/basic_prompt.ini`

`Function Options`: `hy-motion.github.io/prompt/Function_options.ini`

## A7 Task Planning with LLM

After receiving the prompts from 'User Input', the LLM output a hierarchical task graph that contains a series of nodes and actions for accomplishing the task. The task graph is saved in an .xml file and serves as a framework for constructing the Behavior Tree that guides the robot's actions. Below we show the detail of experiments in 'Tasks with human instructions' part of Sec. 4.3. For each task, we present the task graph generated by LLM, and the Behavior Tree constructed from it.

```
Input: Open the drawer and pick up the drill.
```

```
<root BTCPP_format="4" >
 <BehaviorTree ID="MainTree">
 <Sequence name="root">
 <FindObject name="find_object" goal="{DrawerPosition}"/>
 <Fallback>
 <Distance name="object_in_reach" message="{DrawerPosition}"/>
 <MoveTarget name="move_target" message="{DrawerPosition}"/>
 </Fallback>
 <OpenDrawer name="open_drawer"/>
 <ObjectDetect name="object_detect" goal="{DrillPose}"/>
 <SinglePick name="single_arm_pick" message="{DrillPose}"/>
 </Sequence>
 </BehaviorTree>
</root>
```

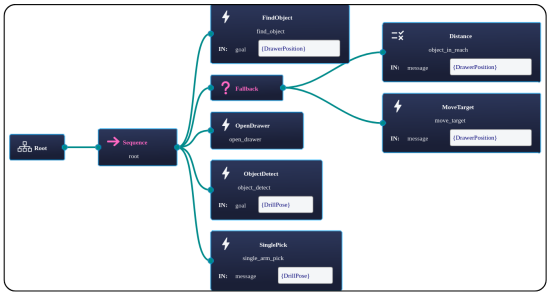


Figure A8: Task planning of 'Open drawer and pick object'.

```
Input: Find the door and open it.
```

```
<root BTCPP_format="4">
 <BehaviorTree ID="MainTree">
 <Sequence name="root">
 <FindObject name="find_object" goal="{DoorPosition}"/>
 <Fallback>
 <Distance name="object_in_reach" message="{DoorPosition}"/>
 <MoveTarget name="move_target" message="{DoorPosition}"/>
 </Fallback>
 <OpenDoor name="door_open"/>
 </Sequence>
 </BehaviorTree>
</root>
```

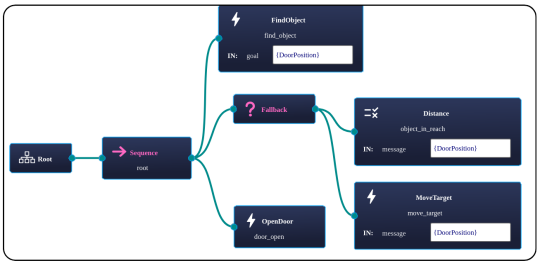


Figure A9: Task planning of 'Approach and open door'.

```
Input: Pick up the cracker and put it into the box.
```

```
<root BTCPP_format="4">
 <BehaviorTree ID="MainTree">
 <Sequence name="root">
 <FindObject name="find_object" goal="{CrackerPosition}"/>
 <Fallback>
 <Distance name="object_in_reach" message="{CrackerPosition}"/>
 <MoveTarget name="move_target" message="{CrackerPosition}"/>
 </Fallback>
 <SinglePick name="single_arm_pick" message="{CrackerPosition}"/>
 <FindObject name="find_object" goal="{BoxPosition}"/>
 <Fallback>
 <Distance name="object_in_reach" message="{BoxPosition}"/>
 <MoveTarget name="move_target" message="{BoxPosition}"/>
 </Fallback>
 <ObjectPlace name="object_place" message="{BoxPosition}"/>
 </Sequence>
 </BehaviorTree>
</root>
```

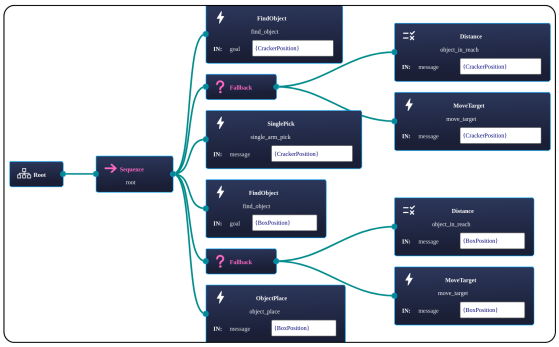


Figure A10: Task planning of 'Pick and place'.

```
Input: Pick up the box and put it on the table.
       ('manipulation_mode_selector'=on)
```

```
<root BTCPP_format="4">
 <BehaviorTree ID="MainTree">
 <Sequence name="root">
 <FindObject name="find_object" goal="{BoxPosition}"/>
 <Fallback>
 <Distance name="object_in_reach" message="{BoxPosition}"/>
 <MoveTarget name="move_target" message="{BoxPosition}"/>
 </Fallback>
 <ObjectDetect name="object_detect" goal="{BoxPose}"/>
 <Fallback>
 <Sequence>
 <WhetherSingleArm name="whether_single_arm" message="{BoxPose}"/>
 <SinglePick name="single_arm_pick" message="{BoxPose}"/>
 </Sequence>
 <DualPick name="dual_arm_pick" message="{BoxPose}"/>
 </Fallback>
 <FindObject name="find_object" goal="{TablePosition}"/>
 <Fallback>
 <Distance name="object_in_reach" message="{TablePosition}"/>
 <MoveTarget name="move_target" message="{TableGeneratorPosition}"/>
 </Fallback>
 <ObjectPlace name="object_place" message="{TablePosition}"/>
 </Sequence>
 </BehaviorTree>
</root>
```

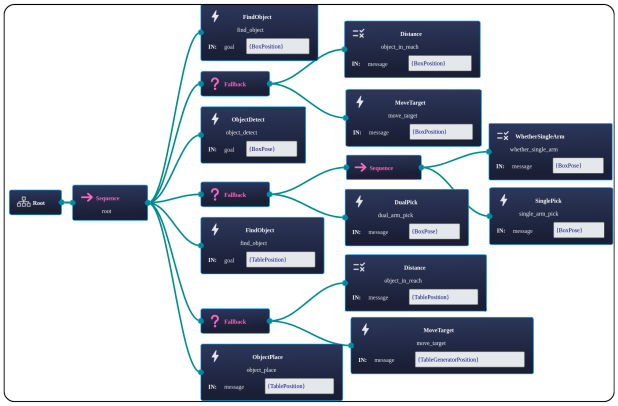


Figure A11: Task planning of 'Dual-arm pick place'.

## A8 Long-horizon Task

### Environment Setup

The AprilTag system [4], which incorporates a vision-driven algorithm, was used during the long-horizon task to identify the relative objects' location and direction of recognized tags. Within the actual environment, we employ AprilTags to gather task-specific observations. A single visual marker on the door allows for the determination of the door handle's relative position. The robot searches for the tag if it doesn't exit the camera's field of view (FOV). Additionally, AprilTags enable the identification of the drawer's relative positions.

We performed the long-horizon shown in Fig. 1. And the task graph for the long-horizon taks generated by LLM can be found in Fig. 7. For the full video, please refer to https://hy-motion.github.io/

## References


[1] A. Laurenzi, D. Antonucci, N. G. Tsagarakis, and L. Muratore. The xbot2 real-time middleware for robotics. *Robotics and Autonomous Systems*, 163:104379, 2023.

[2] A. Laurenzi, E. M. Hoffman, L. Muratore, and N. G. Tsagarakis. Cartesi/o: A ros based real-time capable cartesian control framework. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 591–596. IEEE, 2019.

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

[4] E. Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation*, pages 3400–3407. IEEE, 2011.