LVDG: A VLM-Driven and Graph Theory Framework for Imitation Learning with Dynamic System and Geometric Constraint

Haohui Huang , *Member, IEEE*, Kailun Huang, Yikai Fu, Liting Zeng, Yi Guo, Chenguang Yang, *Fellow, IEEE*

***Abstract*—With the rapid advancement of embodied intelligence, significant progress has been made in environmental perception and autonomous robotic task planning. Although methods based on vision language model (VLM) excel in zero-shot perception in dynamic environments, they struggle to effectively resist unexpected disturbances when planning tasks using traditional large language model (LLM) and behavior trees. This limits their applicability in dynamically adjusting subtask execution sequences or optimizing motion trajectories to cope with disturbances. To address these challenges, this paper proposes a robust dynamic system (DS) learning and planning framework that integrates LLM, VLM, and graph theory (LVDG), to enhance robot autonomy and robustness in complex tasks. To fully leverage the robustness of DS, the framework introduces a skill knowledge library that includes motion behaviors based on Lyapunov dynamic system (LPV-DS) and perception behaviors based on VLM. The LLM dynamically generates behavior graphs based on VLM perception data and robot state, adjusting node sequences to resist unknown disturbances. Additionally, we introduce a pose correction mechanism based on geometric constraints, which restricts the orientation of the end-effector and leverages physical structural constraints to ensure smooth task execution. Experimental results show that, compared to the Inverse Kinematics and OpenVLA model, DS can adaptively modify trajectories under target displacement or external disturbances, ensuring task completion without the need for extensive training data, thereby improving efficiency. The proposed method has been experimentally validated in task planning, trajectory control, and environmental adaptability, demonstrating significantly improved success rates and robustness. It offers a novel solution for the stable execution of embodied intelligent robots in dynamic environments.**

***Note to Practitioners*—The motivation for this work stems from the need for robots to reliably perform complex tasks in dynamic and unpredictable environments, such as warehouses. Traditional task planning methods often struggle to quickly adjust action sequences in the face of sudden changes, leading to task failures. The solution proposed in this paper combines perception, planning, and control to form a flexible framework that enables robots to dynamically adjust their behavior in response to object movement or external disturbances, thus improving task success rates. In particular, the introduction of Lyapunov energy field-based dynamic system control and graph-based behavior planning significantly enhances the robustness of the system. Although the method has shown excellent performance in structured experimental environments, more research is needed for its application in real-world scenarios. However, the framework offers a stable solution for executing tasks in dynamic and complex environments. Future work can focus on expanding the skill library and improving real-time learning capabilities to further improve robot adaptability and reliability in real-world settings.**

***Index Terms*—Dynamic system, Large Language Model, Vision Language Model, Behavior Graph, Skill Knowledge Library, Autonomous Planning, Disturbance Resistance.**

Haohui Huang, Kailun Huang, Yikai Fu, and Liting Zeng are with the School of Automation, Guangdong University of Technology, Guangzhou 528300, China.

Yi Guo is with the School of Automation, Shanghai Jiao Tong University, Shanghai 200240, China.

Chenguang Yang is with the Department of Computer Science, University of Liverpool, L69 3BX Liverpool, U.K (e-mail: cyang@ieee.org).

This work was supported in part by the National Natural Science Foundation of China under Grants 62303120 and 62473251.

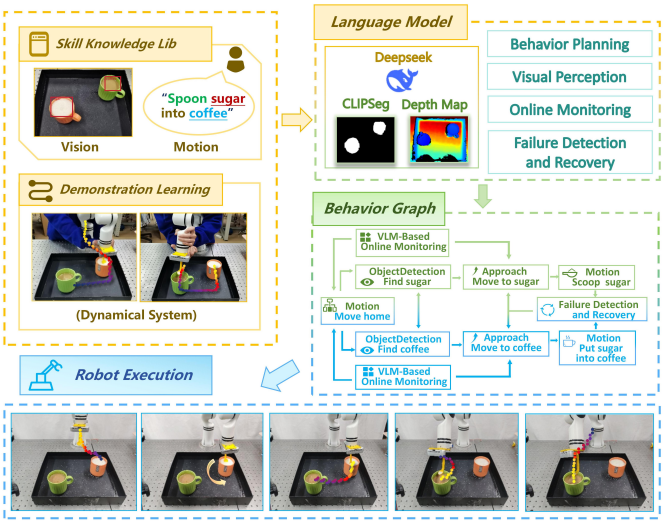

Fig. 1: The skill knowledge library is divided into perceptual behaviors and action behaviors. LLM will learn all the skills in the knowledge library and construct a behavior graph framework, thereby guiding the robot to execute tasks according to human instructions.

I. Introduction

WHEN executing tasks in complex real-world environments, a robot's resilience to interference is crucial, yet challenging—especially when there are sudden changes in environmental conditions. This not only requires the robot to possess robust perception and decision-making capabilities but also the ability to cope with unexpected environmental changes, allowing it to detect and respond to external variations during task execution and autonomously replan an optimal route. Moreover, the robot must understand human task instructions, respond accordingly according to the current environment and its own state, and convert human textual commands into a sequence of robotic actions. This process involves mapping high-level task planning to low-level controllers while continuously monitoring external changes in real time to generate an execution trajectory for the robot.

With the rise of LLM and VLM, robots have gained stronger capabilities in semantic understanding and visual perception. Notably, models like DeepSeek [1] demonstrate flexibility in interpreting instructions and generating structured behavior plans. However, while LLM provide high-level reasoning and task planning, they lack real-time responsiveness to dynamic environmental changes. In contrast, VLM excel at perception and can detect external disturbances as they occur.

In this work, we propose a behavior planning framework that combines LLM, VLM, DS model, and graph theory (LVDG) to address both semantic understanding and physical adaptability. The LLM parses natural language commands and generates a behavior graph (BG), where each node corresponds to a skill selected from a pre-built knowledge base containing both perception and action primitives. When the VLM detects environmental disturbances, it triggers a dynamic update of the behavior graph, enabling the robot to replan its strategy and continue task execution adaptively. As illustrated in Figure 1, the framework achieves a seamless integration from high-level reasoning to low-level control.

The DS model is used for the generation of real-time trajectory. It supports generalization with few demonstrations and is highly suitable for high-dimensional motions. However, for fine-grained manipulation tasks, such as scooping sugar into a cup, the lack of precise orientation control may lead to failure. To mitigate this, we introduce a lightweight pose correction module based on geometric constraints. This module adjusts the orientation of the end effector according to the physical structure and contact geometry, enhancing stability during critical actions without compromising the generality of the DS model.

We summarize the main contributions of this work as follows:

- We propose a robust dynamic system learning and planning framework (LVDG) that combines LLM, VLM, and graph theory to improve robot autonomy and robustness in complex tasks.
- We introduce a skill knowledge library with LPV-DS motion and VLM perception behaviors, where the LLM generates behavior graph and dynamically adjusts the node order based on VLM perception data and the robot's state to resist unknown disturbances.
- We introduce a pose correction mechanism based on structural and geometric constraints, which improves the success rate of the DS model in high-precision tasks.

II. RELATED WORKS

A. VLM and LLM for Motion Planning

Endowing robots with autonomy in highly complex tasks has always been a significant challenge in robotics research. With the gradual development of LLM and VLM, this challenge has seen remarkable breakthroughs, particularly in autonomous planning and execution, opening new opportunities for robotics ([2] [3] provide an overview). Using the semantic comprehension capabilities of LLM and the architecture of BT, human language instructions can be transformed into executable low-level actions, achieving better results in high-level task planning [4] [5] [6] [7]. Meanwhile, the effectiveness of VLM in spatial reasoning and high-level task planning has been increasingly validated [8] [9] [10]. In recent years, Transformer-based planners, such as [11] [12] [13], have played a crucial role in the prediction and planning of robot actions. Some studies have further advanced robot behavior planning by directly using human instructions to guide robot policy codes [14] [15]. However, traditional end-to-end training methods, such as Google's RT-2 [12] and Stanford's OpenVLA [16], rely heavily on extensive training data and expert demonstrations, making the models complex and less portable. In addition, these methods often overlook sudden environmental disturbances and the robot's ability to adapt to environmental feedback during execution, leading to significant discrepancies between high-level planning and actual execution. This study adopts DeepSeek as the primary framework, integrating VLM and depth information as a visual recognition and correction module. In addition, we introduce a disturbance-resistant monitoring mechanism that combines graph theory with VLM. Ultimately, the framework guides the robot to complete tasks by dynamically adjusting the node sequence of the behavior graph.

B. Autonomous Dynamical System Learning

Unlike traditional end-to-end models, this study adopts a motion planning approach based on DS model, which enables efficient learning with only a small number of demonstration samples, thereby avoiding the reliance on extensive data and expert demonstrations typical of conventional methods. In the field of robot motion planning, learning methods based on stable DS have become a cutting-edge research focus [17] [18] [19] [20]. In particular, the method proposed by [21] for learning globally asymptotically stable autonomous dynamical systems (ADS) in two steps, and the approach introduced by [22] for solving DS-based pose generation in three-dimensional space using Riemannian manifolds, have provided significant inspiration for our work. We apply this method to the skill library constructed in this study, serving as the robot's motion skill module, which significantly enhances the robot's spatial generalization capabilities across different tasks. By integrating the DS model, we not only optimize the robot's motion trajectory, but also improve its resilience to external disturbances. Even if the target object in the environment changes, the model can dynamically generate a new trajectory, significantly improving the adaptability and efficiency of the robot in complex environments.

III. AUTONOMOUS ROBOT BEHAVIOR PLANNING

A. Problem Statement

The challenge of enabling robots to autonomously execute tasks based on high-level instructions in complex environments can be described as follows. We assume that a task-specific instruction provided by a human is denoted as i, and that this instruction can be executed by the robot. Additionally, we establish a skill knowledge library in which the robot has access to and can utilize all available skills. This knowledge

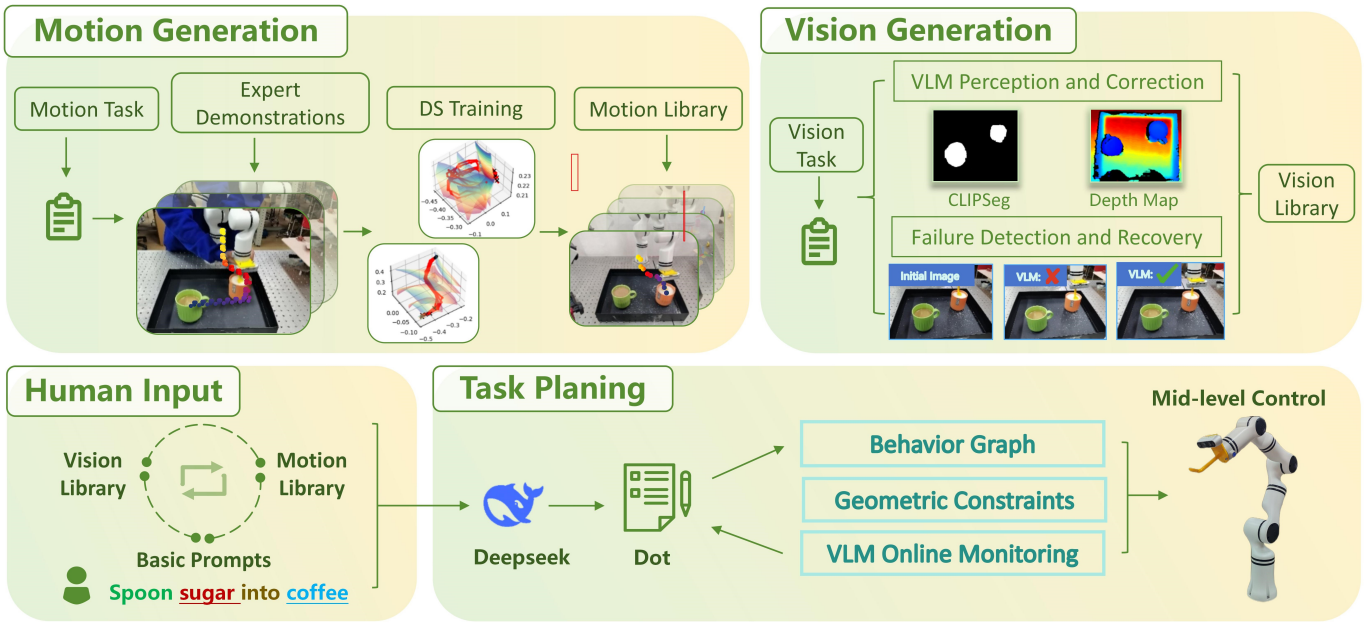

Fig. 2: **Overview of the Framework.** We decompose the framework into four components: **Motion Generation** is responsible for learning and training motion skills for various subtasks (based on DS model) and storing them in the motion library. **Vision Generation** is tasked with identifying and correcting target objects to obtain their coordinates.**Human Input** includes received task instructions and the initialization of the skill library. **Task planning** leverages DeepSeek's reasoning and planning capabilities to generate a behavior graph, which guides the robot's actions and transmits motion commands to the physical robot. Meanwhile, the VLM is utilized to continuously monitor dynamic disturbances in the external environment in real time.

library comprises skills based on VLM (for three-dimensional perceptual behaviors) and LPV-DS (for three-dimensional motion behaviors). Each skill within the knowledge library is associated with its corresponding real-world semantic meaning (e.g., "approach the sugar-containing cup"). The skill library is shown in Table I.

By leveraging LLM, task graph code corresponding to instruction *i* can be directly generated. This task graph code is stored in *dot* format and used to construct a behavior graph. The task graph defines a sequence of skills that the robot must execute in different states to accomplish the human-issued instruction. Each node in the behavior graph represents a specific skill that the robot needs to perform in its current state. Based on the current state of the robot, the behavior graph dynamically guides the execution of appropriate skills to ensure the successful completion of the task.

When an unexpected environmental disturbance occurs, the VLM analyzes the disturbance along with the robot's current state and regenerates the task graph code corresponding to instruction *i*. The framework then dynamically adjusts the node sequence of the behavior graph to enhance task success rates. Through this approach, the robot can effectively understand human instructions and tasks, autonomously invoke skills from the knowledge library, construct a reliable and stable behavior graph, and execute the task accordingly—all while maintaining strong robustness against external disturbances.

B. Overview of the Framework

We propose a robotic behavior planning system that integrates graph theory, LLM, VLM, and DS models to execute complex autonomous motion tasks. This system takes high-level behavioral instructions as input, representing highly intricate tasks. Through the LLM, these high-level instructions (complex tasks) are decomposed into mid-level and low-level instructions, ultimately forming a behavior graph. Figure 2 provides an overview of the system.

First, we establish a skill knowledge library for the REALMAN robotic arm, which consists of LPV-DS-based action behaviors and VLM-based perception behaviors. Each behavior is associated with a corresponding behavior label and an API interface. The behavior labels store the name, type, and detailed semantic description of each behavior. Additionally, prompt information is pre-defined as prior knowledge for the LLM, including input conditions, robot characteristics, and descriptions of available skills.

The prompt information is preloaded into the LLM. When the robot is ready to execute a task, a human operator inputs the task instruction along with all relevant skill labels from the knowledge library. Upon receiving this information, the LLM generates a customized set of behavior graph code based on the task requirements and stores it in *dot* format. This file is then used to generate the behavior graph, which governs the execution of robotic tasks.

Each node in the behavior graph is mapped to a behavior API interface in the skill knowledge library. The behavior graph dynamically guides the robot to execute different actions based on its current state and environmental conditions, ultimately ensuring task completion. Furthermore, in the event of unexpected external disturbances, the VLM analyzes the interference along with the robot's state and regenerates the task graph code, dynamically adjusting the node sequence of

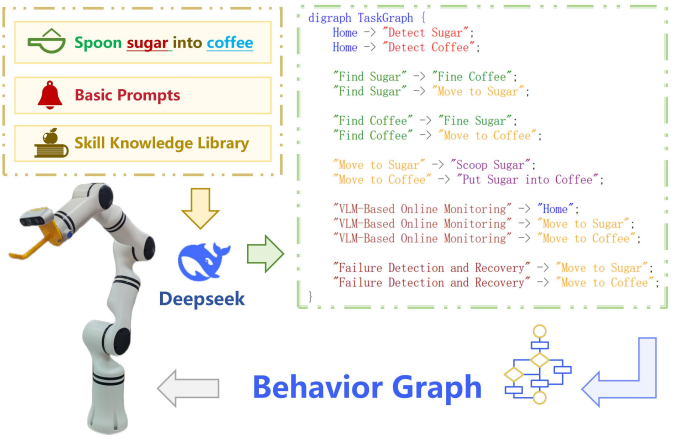

Fig. 3: Behavior Planner Grounding LLM.

the behavior graph to attempt error correction and maintain task success.

Meanwhile, sensor feedback and the robot's current state are continuously integrated into the behavior graph, ensuring coherent execution. The host PC handles high-level planning and behavior graph generation, while the robot controller manages low-level motion execution to complete the task.

C. Demonstration Learning of LPV-DS

In the skill knowledge library, the behavior API interfaces correspond to DS models, each semantically associated with specific mid- and low-level commands. To train multiple DS models, a demonstration dataset $D = \{\xi_{t,n}^k\}_{t=1,n=1,k=1}^{T_n,N,K}$ can be collected, where $\xi_{t,n}$ represents the robot's position at time step t of the n^{th} demonstration trajectory, N is the total number of demonstration trajectories, T_n is the length of the n^{th} trajectory, k corresponds to the k^{th} DS model, and K is the total number of trained DS models. Using dataset D, a primitive attractor DS can be learned through regression algorithms:

$$\dot{\xi} = o(\xi) \tag{1}$$

However, a key challenge is that the learned DS often lacks global asymptotic stability (GAS), which is crucial for ensuring robust and reliable trajectory generation. To ensure that the DS possesses global asymptotic stability (GAS), a correction term u can be incorporated into the DS using a control Lyapunov-based approach:

$$\dot{\xi} = g(\xi) = o(\xi) + u \tag{2}$$

where u is computed online by solving a convex optimization problem:

$$\min_u u^T u \tag{3}$$

subject to the constraint:

$$\left(\frac{\partial V}{\partial \xi}(\xi)\right)^T (o(\xi) + u) \leq -\rho(\xi) \tag{4}$$

$V(x)$ represents the energy function, also known as the Lyapunov function. The constraint ensures that the energy along the trajectory of the DS $\dot{\xi} = g(\xi)$ always decreases, thereby guaranteeing that the trajectory converges to the minimum of the energy function at ξ^*. By representing $V(x)$ using a neural network and imposing constraints on its parameters, we ensure that it has a unique minimum. This energy function serves as the foundation for stability analysis, enabling the DS to be globally asymptotically stable. For further details, refer to [21]. Multiple DS models were trained using the above method, with each model serving as a behavior API interface. By inputting the initial and target coordinates, a smooth trajectory can be generated. Each DS model is semantically linked to specific mid- and low-level commands (e.g., "approach the box" or "grasp the box"). Compared to traditional end-to-end models, DS-based trajectory generation significantly improves success rates and reduces computational costs through few-shot training.

TABLE I: Skill Knowledge Library

Skill Behavior	**Type**	**Skill Tag**
Homing	*action*	'The robotic arm returns to its initial pose'
Approach	*action*	'The robotic arm approaches the {*target object*}'
Grasp	*action*	'The robotic arm grasps the {*target object*}'
Spoon	*action*	'The robotic arm scoops the {*target object*}'
Pour	*action*	'The robotic arm pours the {*current object*}'
Release	*action*	'The robotic arm releases the {*current object*}'
Recovery	*action*	'If VQA detection fails re-execute the previous action node'
Object Detection	*perception*	Detect the {*target object*} and obtain its 3D pose'
Object Correction	*perception*	Correct the 3D pose of the {*target object*}'
Failure Detection	*perception*	'Verify task success using VQA to assess completion status'

D. Language Model for Behavior Planning

1) Skill Knowledge Library: In highly complex environments, it is challenging for robots to autonomously complete intricate tasks. To bridge the gap between semantic instructions and the robot's actual actions, we establish a skill knowledge library where each skill is mastered by the robot. By effectively utilizing the skills within this knowledge library, the robot can accomplish complex tasks. This approach decomposes high-level tasks (abstract commands) into a sequence of sub-tasks composed of various skill combinations (mid- and low-level instructions). This not only enhances the interpretability of the robot's behavior during task execution but also enables the robot to handle increasingly complex tasks as the skill knowledge library expands.

We divide the skill knowledge library into two main components: a three-dimensional action behavior library based on LPV-DS and a three-dimensional perception behavior library based on VLM. The action behavior library controls the movement of the REALMAN robotic arm and is associated with different DS model, each combined with semantic meaning. It is important to note that the trajectories output by these DS model, along with their associated semantics, go beyond simple actions like "grasp" or "lift." Instead, they represent action trajectories that are influenced by environmental feedback and

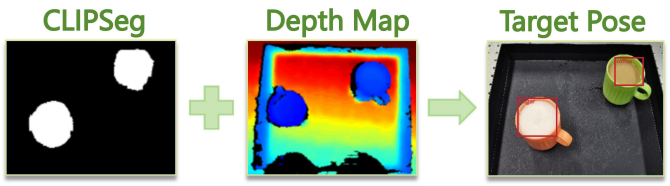

Fig. 4: The CLIPSeg segmentation model within the VLM, combined with depth information, is used to obtain the 3D pose of the target object.

are linked to a sequence of sub-tasks, enabling the execution of highly complex compliant tasks such as scooping sugar into coffee or folding clothes. The action behavior library can be expanded and combined according to the needs of specific motion tasks and interaction requirements in the environment. All skills in the skill knowledge library are shown in Table I.

Perception behavior relies on the VLM and the robot's internal sensors (such as depth cameras, RGB cameras, force sensors, etc.) to detect the 3D pose of objects. By combining this information with the robot's own parameters, the system infers whether the robot can reach the object's location. The design of perception behavior, as well as the types of action behaviors, can be added or modified based on task requirements. Additionally, multiple sensors can be integrated and used in combination to fuse data from various modalities. Meanwhile, we leverage the VLM's dynamic perception capabilities by deploying cameras outside the experimental environment to continuously monitor for unexpected disturbances and assess their impact on the robot's normal operation.

2) LLM and VLM Generated Task Planner: Although LLM can leverage their powerful reasoning and semantic understanding capabilities to guide robotic actions, the responses they generate can be highly varied. The robot cannot directly act on these responses. To achieve the desired output, prior knowledge must be provided to the LLM, including: the hardware parameters of the REALMAN robot, the fundamental concepts in the skill knowledge lib, an introduction to the basic structure of the skill knowledge lib and skill labels, example applications, and finally, the expected output format (such as the concept of behavior graph, node definitions, etc.).

To convert high-level instructions into executable mid- and low-level commands that meet task requirements, we introduce behavior graphs from graph theory as a bridge between LLM-driven high-level planning and the robot's actual motion execution. The use of behavior graphs provides a hierarchical, graphical, and feedback-driven framework for controlling the robot's actions and decision-making process. This framework consists of nodes with different functionalities. In this study, complex instructions are decomposed into behavior graphs composed of multiple nodes. The more complex the instruction, the more branches and feedback mechanisms are generated within the behavior graph. Based on a predefined skill knowledge library, the LLM generates behavior graph code according to the input skills and high-level task instructions, storing the output as a *dot* file. We interact with DeepSeek-V3 via the DeepSeek API, utilizing it as the LLM. The DeepSeek-V3 model directly outputs *dot* files, which are then used to generate behavior graphs, enabling efficient task planning.

For example, in the task of scooping sugar into coffee, the robot ultimately follows the behavior graph to complete the task, where each subtask corresponds to a node in the task graph. The execution process begins with identifying the coffee and sugar, retrieving their respective coordinates. The sugar's coordinates are then fed into the "Approach Sugar" DS model to generate a motion trajectory. After the scooping action is performed, the system verifies whether it was successful. Next, the coffee's coordinates are input into the "Approach Coffee" DS model, generating another trajectory. Finally, after pouring the sugar into the coffee, the system revalidates the success of the action. (For a detailed explanation, please refer to the Experimental Section)

3) VLM Localization and Correction: For the localization and correction of the target object, we employ the VLM recognition method, using the CLIPSeg segmentation model to obtain the target image mask. This mask is then combined with depth information to determine the 3D pose of the object. Specifically, we perform morphological processing on the image mask to obtain a circular mask with the target object's center as the origin. This circular mask is then used in conjunction with depth information to derive the 3D pose. The depth camera and robotic arm have been pre-calibrated, ensuring that the target coordinates are ultimately transformed into the base coordinate system. The localization process is illustrated in Figure 4.

The target pose and the current pose of the robot are input into the next node (via the DS model) to generate the trajectory. The robot then moves along the trajectory to the target object's position, denoted as p_1. To improve task success rates, we call the VLM again at the p_1 position to re-acquire the target object's pose and adjust the robot's position accordingly. This method fully leverages the potential of VLM for both near and far-range recognition, ensuring the accuracy of the object's pose.

E. End-Effector Pose Correction Based on Geometric Constraints

To improve the success rate and stability of spoon insertion in sugar-scooping tasks, we introduce a lightweight pose correction strategy on top of the trajectory generated by the DS model. This strategy leverages physical structure constraints by applying geometric restrictions to the end-effector orientation, enabling fine-grained compensation without modifying the original DS framework. It ensures that the spoon can smoothly enter the cup opening, avoid collisions, and stably complete the scooping action.

*1) **Perception-Based Estimation of Cup Geometry**:* To define the geometric constraints, we first estimate the spatial center p_c and the radius r_c of the opening of the cup. We use OWL-VIT for object detection and SAM for mask extraction to segment the visible rim of the cup. From the binary mask, three key points are extracted:

- the leftmost and rightmost rim pixels (used to calculate the radius r_c).
- the pixel closest to the robot in depth (used to determine the cup center p_c).

Fig. 5: The left image shows the output of OWL-VIT combined with SAM, where the yellow region indicates the extracted mask and the red outline marks the cup rim. The right image presents a close-up view of the sugar-scooping action.

TABLE II: NOMENCLATURE

p_c	Position of the center point of the cup.
p_s	End-point trajectory output by DS model.
z_s	The direction vector of the spoon.
R_s	Initial end-effector orientation matrix.
l_s	Length of the spoon.
r_c	Radius of the cup opening.
θ_{max}	Maximum allowable orientation error angle.
ϵ	Collision-free safety boundary.
$\vec{n}_{ideal}$	Ideal insertion direction.

The 3D cup center p_c is estimated by fusing the horizontal midpoint and the closest point using the depth map. The radius r_c is derived in a similar way from the projected edge of the rim. All points are represented in the world coordinate system. For an accurate comparison of orientation, we define a local cup frame $\mathcal{F}_{\text{cup}}$ centered at p_c, whose z axis is aligned with the normal surface (typically upward or vertical in the setup). As shown in Figure 5.

*2) **Ideal insertion direction**:* Considering that the ideal insertion direction in scooping tasks is vertically downward, we define:

$$\vec{n}_{\text{ideal}} = [0, 0, 1]^T \tag{5}$$

This vector represents the direction of the target for the correction of the pose.

*3) **Pose misalignment angle**:* The angular deviation between the current end-effector direction $\vec{z}_s = \frac{p_c - p_s}{\|p_c - p_s\|}$ and the ideal insertion direction is calculated as:

$$\theta = \arccos\left(\vec{z}_s \cdot \vec{n}_{\text{ideal}}\right) \tag{6}$$

*4) **Insertion depth constraint**:* The insertion depth from the spoon tip to the center of the cup must not exceed the usable length of the spoon:

$$\|p_c - p_s\| \leq l_s - \delta \tag{7}$$

where δ is a safety margin (typically 2.5–3 cm).

*5) **Horizontal offset constraint**:* The lateral projection of the spoon tip on the plane orthogonal to the insertion direction should be limited to avoid collision:

$$\left\| \left(I - \vec{n}_{\text{ideal}} \vec{n}_{\text{ideal}}^T\right) (p_s - p_c) \right\| < r_c - \epsilon \tag{8}$$

This ensures that the spoon remains near the centerline of the cup opening and avoids scraping the rim.

*6) **Pose misalignment check**:* If $\theta > \theta_{\text{max}}$ or other constraint conditions are not satisfied, the pose is considered misaligned and triggers the correction procedure. If the correction still does not satisfy the constraint, the system will re-execute the skills of identifying and approaching the sugar to ensure the task can proceed correctly.

*7) **Rotation axis computation**:* The axis of rotation between the current direction z_s and the ideal direction $\vec{n}_{\text{ideal}}$ is defined as:

$$r_{\text{axis}} = \vec{z}_s \times \vec{n}_{\text{ideal}} \tag{9}$$

*8) **Correction matrix construction**:* To avoid abrupt changes in the pose correction process, we introduce a scaling factor λ into the Rodrigues rotation formula [23]. The corrected rotation matrix is constructed as:

$$R_{\text{corr}} = I + \sin(\lambda\theta)[\hat{r}]_\times + (1 - \cos(\lambda\theta))[\hat{r}]_\times^2 \tag{10}$$

where $\hat{r} = \frac{r_{\text{axis}}}{\|r_{\text{axis}}\|}$ is the unit rotation axis, $[\hat{r}]_\times$ is the skew-symmetric matrix of $\hat{r}$, and $\lambda \in (0, 1]$ is a scaling factor. For example, $\lambda = 0.5$ means that the correction only covers half of the angular deviation, allowing for a smoother and more stable pose adjustment.

During the adjustment process, the end effector is not immediately aligned with the ideal normal n_{ideal}, but gradually corrected based on the continuous trajectory generated by the DS. This strategy offers the following advantages:

- It preserves the temporal continuity and smoothness of the DS-generated trajectory, avoiding system instability caused by abrupt orientation changes;
- The correction matrix acts as a soft constraint, guiding the end-effector orientation to gradually converge toward the desired direction, enhancing robustness and naturalness while ensuring task success.

*9) **Apply correction to final pose**:* The corrected pose is obtained by applying the rotation correction to the original pose:

$$R_s' = R_{\text{corr}} \cdot R_s \tag{11}$$

If using Euler angles for control, the correction can be applied as:

$$\text{rpy}' = \text{rpy} + \delta\text{rpy} \tag{12}$$

F. Adaptive Disturbance Handling Based on Graph Theory and VLM

To address unexpected disturbances in the external environment, we introduce an adaptive disturbance mechanism based on graph theory and VLM. We propose the concept of a behavior graph, where each node represents a skill from the skill knowledge library. By logically sequencing these skills according to task requirements, the robot can effectively accomplish complex tasks. Within this framework, VLM serves as an external monitor, continuously perceiving environmental changes and evaluating task execution. Specifically, this involves two key aspects:

*1) **Evaluation of Key Subtasks**:* VLM utilizes a Visual Question Answering (VQA) mechanism to assess key nodes, determining whether critical subtasks have been successfully completed.

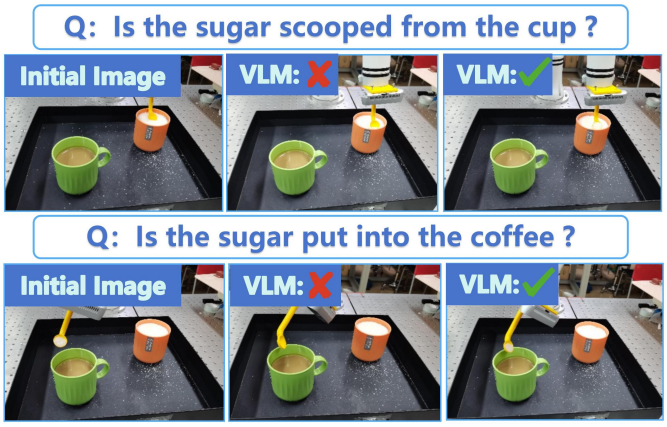

Fig. 6: We use VLM perception skills for failure detection. By inputting images before and after scooping, the system uses VQA reasoning to determine task success.

*2) **Real-Time Environmental Monitoring**:* VLM continuously tracks environmental changes, detecting unexpected disturbances and assessing whether the sequence of nodes in the behavior graph needs to be readjusted to ensure successful task execution.

For dynamic environmental monitoring, during task execution, the VLM continuously captures and analyzes environmental data to enable comprehensive perception and real-time monitoring of the external environment. The primary objective is to detect potential unexpected disturbances that may affect task execution, such as unintended movement of target objects, occlusions, lighting variations, dynamic environmental changes, or other external factors. Upon detecting a disturbance, the system performs an integrated analysis of the current environmental state, robotic motion trajectory, and task progress to dynamically evaluate whether the sequence of nodes in the behavior graph needs to be adjusted to accommodate the new conditions. If the detected disturbance disrupts normal task execution, the VLM triggers an adaptive behavior adjustment mechanism, intelligently modifying the task plan and reordering the behavior graph's nodes to ensure that the robot can autonomously adapt to environmental changes, effectively respond to external disruptions, and ultimately complete the assigned task successfully. As shown in Figure 7. This adaptive planning capability significantly enhances the robot's robustness and autonomous decision-making in complex and uncertain environments, improving overall task reliability and execution efficiency.

For key subtask evaluation, in our predefined task of "scooping sugar into coffee", scooping sugar and pouring sugar are designated as critical nodes. The robot captures images using an external camera and poses the question: "Has the sugar been successfully scooped?" The VLM then responds via VQA with either "Yes" or "No". If the VLM responds "Yes", the robot proceeds to the next node in the behavior tree. If the VLM responds "No", a recovery node is triggered, prompting the robot to recall the previous node, re-execute the scooping sugar behavior API, and repeat the VQA process until the task is successfully completed or the maximum retry limit is reached. During the execution of the behavior graph, each

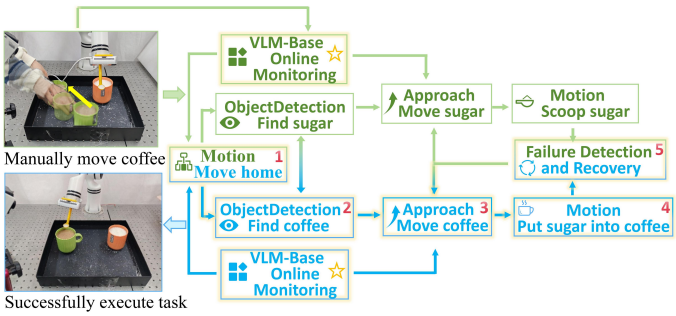

Fig. 7: We use VLM perception skills for environmental monitoring, adjusting behavior graph nodes in real time to re-execute the task.

failure detection node returns two signals: success or failure. As shown in Figure 6. The behavior graph uses these signals to trigger the recovery mechanism. With this approach, the robot will attempt the task multiple times, with each execution following a different trajectory (since the initial and final poses input into the DS model are dynamically generated by the VLM), greatly improving the task success rate.

We deploy the LLM locally without any fine-tuning, integrate it with the REALMAN robotic arm, and evaluate its performance in executing complex tasks under the behavior graph framework. The experimental results validated LLM's capability in the behavior planning stage and further confirmed that VLM serves as the core of robotic perception, significantly enhancing environmental awareness. This study replaces the traditional behavior tree framework with a behavior graph framework and substitutes end-to-end models with DS models, allowing for a performance comparison with conventional approaches. Furthermore, to assess the adaptability of our method in dynamic environments, we introduced external disturbances and analyzed the system's task success rate and robustness.

IV. EXPERIMENT AND EVALUATION

A. Experiment Setup

We placed two cups on the desktop: one containing coffee and the other containing sugar. The testing environment is situated on a laboratory desktop, with the cups randomly placed on the surface. The REALMAN robotic arm has 6 degrees of freedom and is equipped with a depth camera on its end effector. An additional RGB camera is installed in the external environment. The experimental setup is shown in Figure 8. Real-time communication between the robotic arm's low-level drivers and the behavior graph nodes is achieved through the behavior API interface. To facilitate the exchange of information between the LLM, skill knowledge library, behavior tree, and the robotic arm, we utilized the Robot Operating System (ROS). The system has been validated in a real-world environment.

B. The "Scooping Sugar into Coffee" Experiment

We first tested the behavior planning capability of this framework in highly complex robotic tasks. In the experiment, cups containing sugar and coffee were randomly placed, and

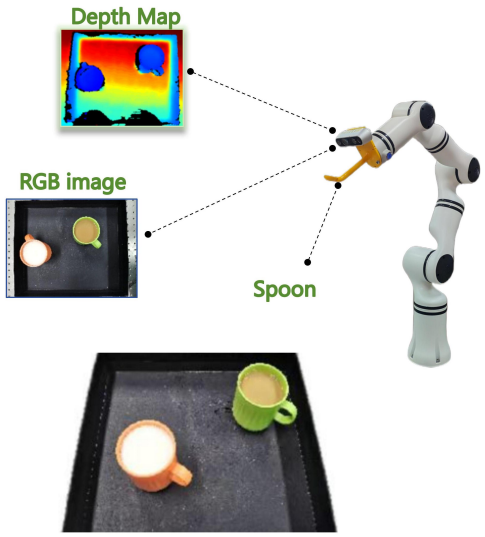

Fig. 8: Experiment setup

the spoon at the end of the robotic arm was required to scoop sugar from one cup and pour it into the coffee cup. Based on the principle illustrated in Figure 3, the high-level task instruction "scoop sugar into the coffee" was input into Deepseek, which then generated an *dot* file to construct a behavior graph. Each node in the behavior graph invoked the corresponding skill from the skill library, thereby guiding the robot to complete the task.

We conducted 20 experiments with randomized object positions to assess the spatial generalization capabilities of the DS model. DeepSeek consistently generated behavior graphs with coherent logic and reliable decision-making. The perception nodes, particularly for identifying sugar and coffee, achieved high accuracy and success rates. Corresponding action nodes, guided by the visual perception outputs, produced motion trajectories that adapted well to positional changes in the environment.

To further improve task execution robustness, we integrated a VLM-based visual correction mechanism before each scooping and pouring action. When the robotic arm approached a target, the visual recognition node was re-triggered to correct the end-effector's posture, enhancing the reliability of these precision-critical operations. Figure 10 shows an example behavior graph generated by DeepSeek, and Figure 9 illustrates the experimental workflow.

To emphasize the role of keypoint correction and perception feedback, we modified the high-level task instruction to: "Scoop sugar into the coffee while performing keypoint detection during task execution." The scooping and pouring actions were designated as critical nodes, where the VQA mechanism was explicitly activated for posture verification and correction. As summarized in Table V, these enhancements notably improved success rates and reduced error propagation across the task pipeline.

C. Comparison of Behavior Graph and Behavior Tree in Task Planning

This section compares the performance of the behavior graph framework and the traditional behavior tree framework

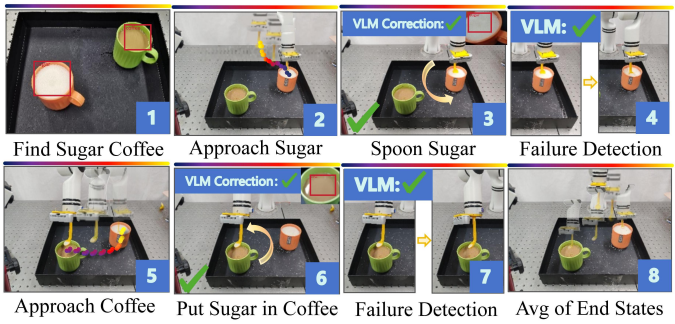

Fig. 9: Experimental Workflow

in task planning, analyzing their differences in task execution efficiency, adaptability, and robustness. To evaluate the adaptability of both methods in dynamic environments, we designed a series of unexpected disturbance tests and observed whether the robot could successfully complete the assigned task.

We first adopted the behavior tree framework [14], where the instruction "scoop sugar into coffee" and the skill knowledge library were provided as inputs. The LLM generated an XML file to construct a behavior tree, where each node represents a skill from the knowledge library. The robot executed actions sequentially according to the node order in the behavior tree. In the absence of external disturbances, the robot is able to complete the task smoothly. However, experimental results indicate that when disturbances are present, the robot can still accomplish the task, but often requires multiple redundant or ineffective actions.

To address this limitation, we improved the behavior tree structure by introducing a behavior graph framework and designed a set of disturbance validation scenarios, including: 1) unexpected movement of the sugar before scooping, 2) unexpected movement of the coffee cup before pouring, and 3) Manual external force disturbs the robotic arm. Under the behavior graph framework, VLM continuously monitors environmental changes in real time. When a significant deviation in object positions or the robotic arm's state is detected, the system dynamically reorders the nodes in the behavior graph based on the robot's completed subtasks and the current environmental changes. This ensures that the robot re-executes necessary steps to successfully complete the task. Figure 10 presents the comparative experiment, while Table IV provides a comparison of task execution times under different disturbances.

D. Comparative Experiment on Robotic Motion Trajectory Generation

This section analyzes the performance of the DS model and end-to-end models in generating robotic motion trajectories, with a focus on trajectory interpretability, training sample size, task execution time, and adaptability to external disturbances. To systematically evaluate the effectiveness of different approaches, we sequentially test Inverse Kinematics (IK), OpenVLA model, DS model, progressively highlighting the advantages of the DS model in complex tasks. Each approach is evaluated through 20 independent experiments, with the target object's position varying in each trial to assess

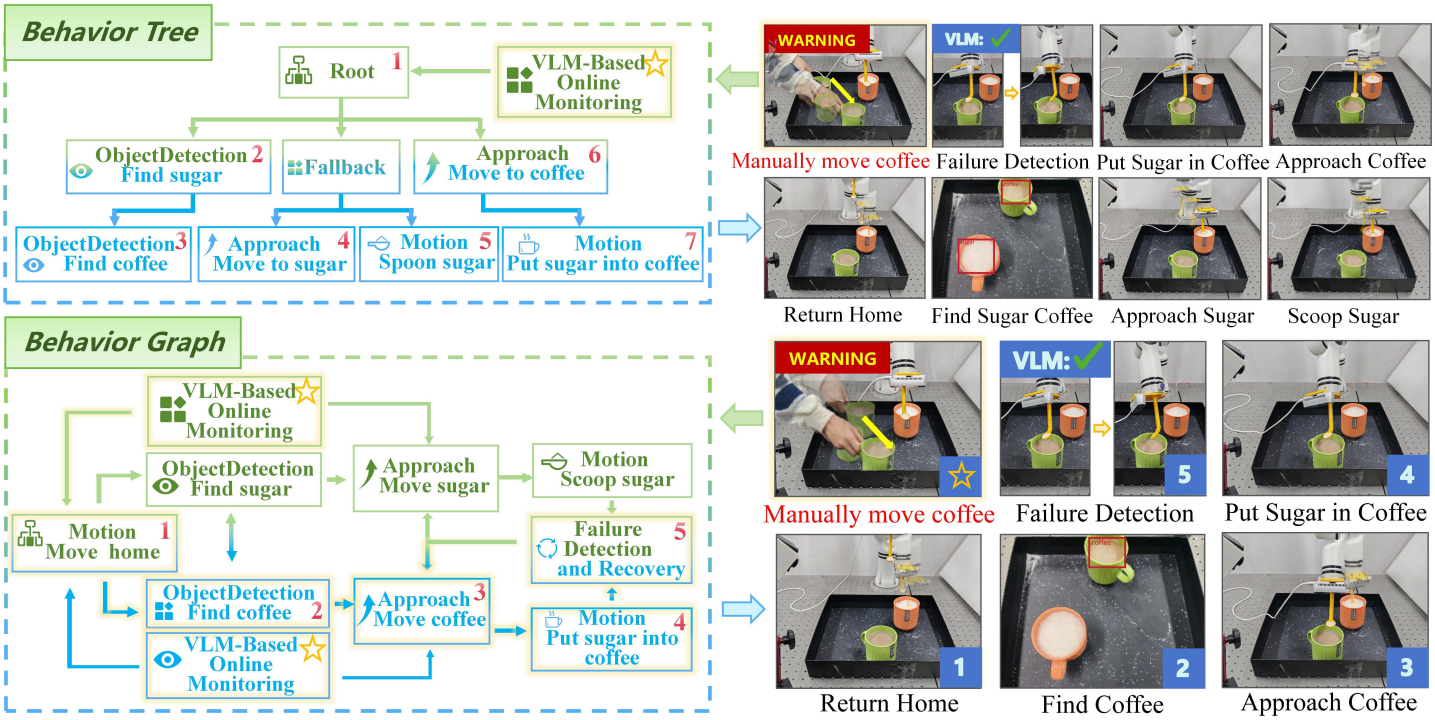

Fig. 10: Comparison of Behavior Graph and Behavior Tree in Task Planning. When the robotic arm successfully scoops the sugar and moves toward the coffee, if the coffee is manually relocated, the behavior graph can dynamically reorder its nodes to quickly complete the task. In contrast, behavior tree cannot recover as efficiently.

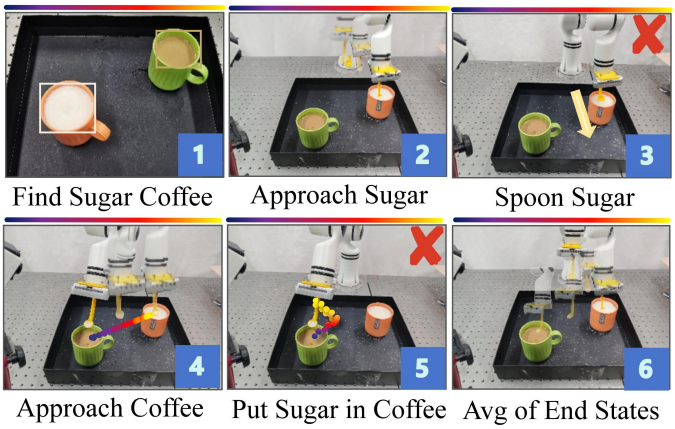

Fig. 11: Inverse kinematics is relatively rigid, making it incapable of performing compliant actions such as scooping and pouring sugar.

the adaptability and stability of different methods during task execution. The experimental results are presented in Table V.

*1) **Experimental Evaluation of IK Method**:* First, we conducted experiments using the IK method within the behavior graph framework proposed in this study. IK generates motion trajectories by computing the joint angles of the robotic arm, enabling the robot to approach the sugar and coffee with relatively fast movement speed. However, since IK primarily relies on analytical kinematic solutions, its motion control tends to be rigid, making it incapable of executing highly dexterous operations such as scooping and pouring sugar. Additionally, when external forces are applied to perturb the robotic arm, IK not only requires re-solving the inverse kinematics equations but also fails to adaptively adjust its trajectory, ultimately leading to task interruption or unstable execution. The experimental procedure is shown in Figure 11.

*2) **Experimental Evaluation of OpenVLA Model**:* Next, we evaluated the OpenVLA end-to-end model, which is a vision language action model. OpenVLA directly predicts the joint angles of the robotic arm using a neural network, eliminating the need for analytical computation as required by the IK method. However, experimental results show that this approach can only perform basic motions such as approaching the sugar and coffee under normal conditions but fails to execute more dexterous actions like scooping and pouring sugar. In addition, separate instructions are required to approach the sugar and approach the coffee.

When external disturbances occur, such as changes in the positions of the sugar or coffee, OpenVLA can only adjust the robotic arm to approach the new target but cannot adaptively modify its trajectory to complete the scooping or pouring actions, leading to task failure. Furthermore, since OpenVLA relies on deep learning, it requires a large amount of training data, operates at a relatively slow speed, and exhibits low overall execution efficiency. The experimental procedure is shown in Figure 12.

*3) **Experimental Evaluation of DS Model**:* Finally, we evaluated the behavior graph and DS model in our proposed approach, where motion trajectories are generated based on the Lyapunov energy field. Compared to IK and OpenVLA, the DS model exhibits greater trajectory interpretability, ensuring global convergence in high-dimensional spaces while enabling real-time trajectory adjustments in dynamic environments. When external disturbances occur, such as changes in the

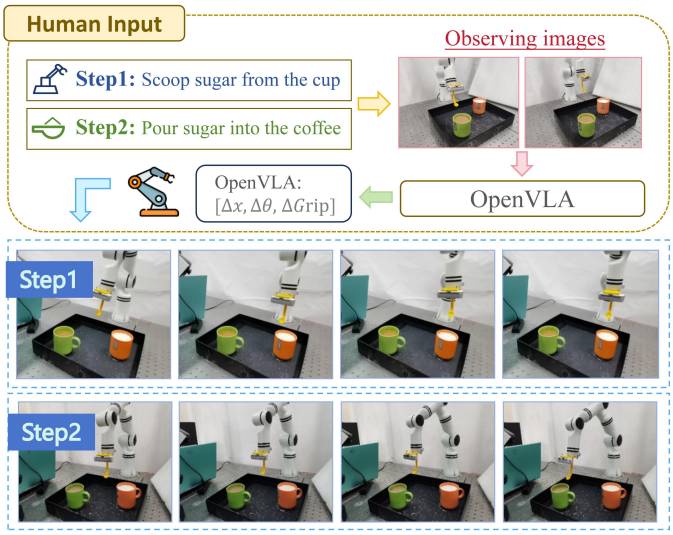

Fig. 12: The execution performance of OpenVLA.

TABLE III: Comparison of Multimodal Data Duration and Training Time Across Different Models.

Training Overhead	IK	Octo	OpenVLA	DS
Multimodal Duration	0	40000h	30000h	**150s**
Training Times	0	18d (8*TPU)	12d (64*A100)	**45s (1*3050)**

positions of the sugar or coffee, the DS model can adaptively regenerate motion trajectories based on the new target position and energy gradient. Additionally, when external forces are applied to perturb the robotic arm, the DS model can dynamically adjust its trajectory according to the current robot position and energy field, ensuring that the robot remains stable and converges to the intended target even after deviating from the planned path.

This energy-field-based control strategy allows the robot to maintain continuous and stable motion in complex environments. Even under disturbances, the DS model can dynamically refine its trajectory, ensuring the successful execution of dexterous operations such as scooping and pouring sugar, thereby improving task success rates and system robustness. As shown in Figure 13.

Experimental results demonstrate that the DS model outperforms IK and OpenVLA in terms of trajectory interpretability, adaptability, and task stability. By leveraging the Lyapunov energy field, the DS model ensures global trajectory convergence and enables real-time trajectory adjustments in response to environmental changes or external disturbances, thereby improving task success rates and system robustness. Moreover, the DS model is lightweight, with low computational overhead, making it suitable for deployment on resource-constrained platforms. Quantitative comparisons are provided in Table III.

E. Comprehensive Evaluation of Pose Correction under Structural Variations

To comprehensively evaluate the robustness and generalizability of the pose correction module, we conducted systematic experiments across multiple physical configurations, including:

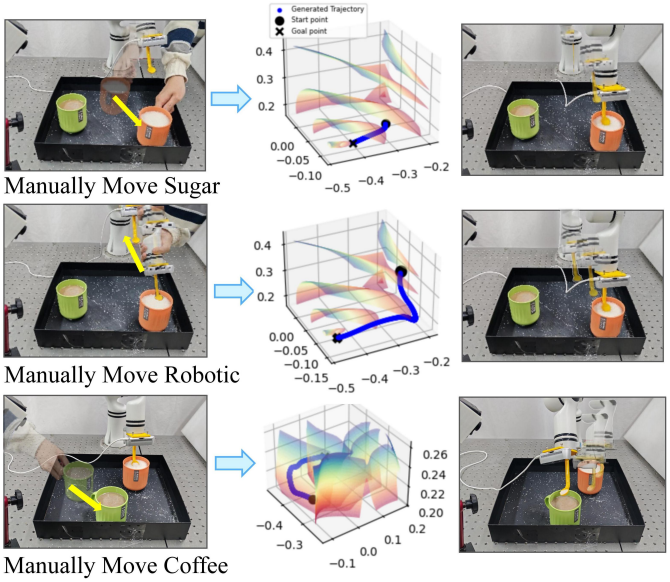

Fig. 13: Performance of the DS Model in Three Anti-Interference Experiments.

TABLE IV: Comparison of Recovery Performance Under Different Disturbances.

Disturbance	Recovery Rate (%)	Adjustment Time (s)
Move Robot Arm	89%	2.87s
Move Sugar	91%	3.24s
Move Coffee	88%	3.09s

- Spoon Lengths: short spoon ($l_s = 8\,\mathrm{cm}$), long spoon ($l_s = 12\,\mathrm{cm}$);
- Spoon Shapes: straight handle, curved handle;
- Cup Types: cylindrical cup, conical tall cup.

This results in 8 test combinations. For each configuration, we compare:

- DS only: Direct DS trajectory without correction;
- Ours: DS with post-processed pose correction module.

Each configuration is tested in 20 randomized runs. We record the scooping success rate and the average angular deviation from the ideal insertion direction. Results are summarized in Table VI.

The results show that the proposed correction module consistently improves alignment and success rate across all structural configurations, with angular deviation reduced by over 15° and success rate increased by around 10%, highlighting its robustness and general applicability.

F. Results Analysis

In the experiment, we evaluated the behavior planning capability of LLM in highly complex tasks and applied it to the REALMAN robotic arm. By using the skill knowledge library as prior knowledge for LLM and inputting high-level task instructions, LLM is able to generate corresponding behavior plans, represented as behavior graph structures, resulting in higher planning success rates and task execution rates.

The experimental results show that this method significantly outperforms the IK method and the OpenVLA end-

TABLE V: Comparison of Success Rate and Execution Time of Inverse Kinematics, OpenVLA, and Dynamic Systems Across Different Subtasks.

Subtask	**Inverse Kinematics**		**OpenVLA**		**Dynamical Systems**	
	Success Rate (%)	**Time (s)**	**Success Rate (%)**	**Time (s)**	**Success Rate (%)**	**Time (s)**
Find sugar	96%	0.35s	97%	0.32s	99%	0.31s
Find coffee	98%	0.34s	97%	0.35s	100%	0.34s
Approach sugar	78%	5.11s	83%	12.3s	**94%**	**4.25s**
Approach coffee	65%	4.32s	84%	13.7s	**92%**	**3.98s**
Scoop sugar	18%	2.64s	×	×	**88%**	**1.98s**
Put sugar into coffee	21%	3.15s	×	×	**86%**	**2.76s**
Scoop sugar (VQA)	26%	3.27s	×	×	**95%**	**2.54s**
Put sugar into coffee (VQA)	29%	3.98s	×	×	**97%**	**3.41s**

TABLE VI: Evaluation of Pose Correction under Different Tool and Container Configurations.

Tool Length		**Tool Shape**		**Container Type**		**Success Rate (%)**		**Orientation Error (°)**		**Improvement**
Short	Long	Straight	Curved	Cylindrical	Conical	DS Only	+ Correction	DS Only	+ Correction	(Success / Error)
✓		✓		✓		82%	**94%**	24.1°	**7.5°**	+12% / ↓16.6°
✓			✓	✓		78%	**90%**	27.3°	**9.2°**	+12% / ↓18.1°
	✓	✓		✓		85%	**95%**	21.8°	**6.3°**	+10% / ↓15.5°
	✓		✓	✓		80%	**91%**	26.5°	**8.8°**	+11% / ↓17.7°
✓		✓			✓	79%	**92%**	25.7°	**8.5°**	+13% / ↓17.2°
✓			✓		✓	75%	**88%**	28.1°	**9.9°**	+13% / ↓18.2°
	✓	✓			✓	84%	**96%**	22.2°	**6.0°**	+12% / ↓16.2°
	✓		✓		✓	79%	**90%**	27.6°	**8.6°**	+11% / ↓19.0°

to-end model in terms of running efficiency, robustness, task success rate, and resistance to disturbances. While IK and OpenVLA can approach the target objects, they still fail to perform dexterous tasks such as scooping and pouring sugar. In contrast, the DS model, based on the Lyapunov energy field, can dynamically adjust the motion path in high-dimensional space, achieving dexterous control tasks with interpretability.

Regarding robustness and disturbance resistance, the DS model demonstrates superior adaptability. When the positions of sugar/coffee containers are altered or the robotic arm is manually displaced, the DS model can autonomously adjust motion trajectories based on updated target positions and real-time states to ensure task completion. In contrast, both IK methods and OpenVLA models exhibit limited path adaptability under external disturbances, resulting in task failure. Furthermore, traditional behavior tree approaches struggle to rapidly reconfigure task node sequences during sudden disturbances, leading to mission failure. The proposed methodology, through dynamic behavior graph adjustment integrated with DS-based trajectory optimization, maintains high task success rates even in the presence of external disturbances.

For geometric constraint correction, the experimental results show that adding the pose correction module significantly improves task success rates and orientation accuracy across different physical configurations. Specifically, with variations in spoon length, shape, and cup type, the pose correction effectively reduces the angular deviation in insertion direction, demonstrating the module's robustness and adaptability in complex environments.

By combining behavior graph and DS model, this method demonstrates higher efficiency, robustness, and task success rates in task planning within complex environments, especially in scenarios requiring high-precision control and resistance to disturbances.

V. Conclusion

This paper presents a robust task planning framework that integrates LLM, VLM, graph-based behavior planning, and DS trajectory generation. The proposed system demonstrates strong autonomy and adaptability, enabling robots to understand high-level human instructions while responding to real-time environmental disturbances.

By incorporating a skill knowledge library, the framework allows LLM to generate behavior graph for structured task decomposition, while the VLM continuously monitors the scene and triggers dynamic updates of the behavior graph in response to unexpected changes. This mechanism ensures that the robot can adjust its execution strategy to maintain task success under interference. In motion generation, the DS model offers high flexibility and generalization with few demonstrations. However, for precise manipulation tasks, such as scooping sugar, it may produce unstable end-effector orientations.

To address this limitation, we introduce a lightweight pose correction module based on geometric constraints. By estimating the cup's rim geometry using OWL-ViT and SAM, and applying direction, depth, and lateral offset constraints, the module corrects the final orientation while preserving DS trajectory structure. Experimental results show that our method significantly outperforms Inverse Kinematics and OpenVLA in success rate, efficiency, and robustness, especially in dynamic and precision-critical scenarios.

In conclusion, this work offers a unified framework that effectively bridges semantic reasoning, perception, and low-level control. It provides a promising solution for stable and intelligent task execution in embodied robots operating in real-world environments.

REFERENCES

[1] DeepSeek-AI, A. Liu, B. Feng, B. Xue *et al.*, "Deepseek-v3 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2412.19437

[2] A. K. Kovalev and A. I. Panov, "Application of pretrained large language models in embodied artificial intelligence," in *Doklady Mathematics*, vol. 106, no. Suppl 1. Springer, 2022, pp. S85–S90.

[3] Y. Dang, K. Huang, J. Huo, Y. Yan, S. Huang, D. Liu, M. Gao, J. Zhang, C. Qian, K. Wang *et al.*, "Explainable and interpretable multimodal large language models: A comprehensive survey," *arXiv preprint arXiv:2412.02104*, 2024.

[4] K. Valmeekam, S. Sreedharan, M. Marquez, A. Olmo, and S. Kambhampati, "On the planning abilities of large language models (a critical investigation with a proposed benchmark)," 2023.

[5] T. Silver, S. Dan, K. Srinivas, J. B. Tenenbaum, L. P. Kaelbling, and M. Katz, "Generalized planning in pddl domains with pretrained large language models," 2023.

[6] L. Guan, K. Valmeekam, S. Sreedharan, and S. Kambhampati, "Leveraging pre-trained large language models to construct and utilize world models for model-based task planning," 2023.

[7] J. Wang, A. Laurenzi, and N. Tsagarakis, "Autonomous behavior planning for humanoid loco-manipulation through grounded language model," 2024.

[8] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," 2023.

[9] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine, "Robotic control via embodied chain-of-thought reasoning," 2024.

[10] T. Yoneda, J. Fang, P. Li, H. Zhang, T. Jiang, S. Lin, B. Picker, D. Yunis, H. Mei, and M. R. Walter, "Statler: State-maintaining language models for embodied reasoning," 2023.

[11] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[12] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[13] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models," *arXiv preprint arXiv:2310.08864*, 2023.

[14] N. Wake, A. Kanehira, J. Takamatsu, K. Sasabuchi, and K. Ikeuchi, "Vlm-driven behavior tree for context-aware task planning," 2025.

[15] F. Ahmad, J. Styrud, and V. Krueger, "Addressing failures in robotics using vision-based language models (vlms) and behavior trees (bt)," 2024.

[16] M. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[17] D. Layeghi, S. Tonneau, and M. Mistry, "Neural lyapunov and optimal control," 2023.

[18] W. Wang, C. Zeng, H. Zhan, and C. Yang, "A novel robust imitation learning framework for complex skills with limited demonstrations," *IEEE Transactions on Automation Science and Engineering*, pp. 1–13, 2024.

[19] S. Mohammad Khansari-Zadeh and A. Billard, "Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 752–765, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889014000372

[20] H. Huang, Y. Guo, G. Yang, J. Chu, X. Chen, Z. Li, and C. Yang, "Robust passivity-based dynamical systems for compliant motion adaptation," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 6, pp. 4819–4828, 2022.

[21] Z. Jin, W. Si, A. Liu, W.-A. Zhang, L. Yu, and C. Yang, "Learning a flexible neural energy function with a unique minimum for globally stable and accurate demonstration learning," *IEEE Transactions on Robotics*, vol. 39, no. 6, pp. 4520–4538, 2023.

[22] S. Sun and N. Figueroa, "Se(3) linear parameter varying dynamical systems for globally asymptotically stable end-effector control," 2024.

[23] A. G. Valdenebro, "Visualizing rotations and composition of rotations with rodrigues' vector," 2016.

Haohui Huang received the Ph.D. degree in Control Science and Engineering from South China University of Technology, Guangzhou, China, in 2020. He was a postdoctoral with School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, from 2020 to 2022. He is currently a lecture with the School of Automation, Guangdong University of Technology, Guangzhou, China. His research interests include intelligent control, imitation learning, and embodied intelligence.

Kailun Huang is currently pursuing the B.S. degree in Automation from Guangdong University of Technology, Guangzhou, China. His research interests include imitation learning, and embodied intelligence.

Yikai Fu is currently working toward the B.E. degree in the School of Data Science and Big Data Technology, Guangdong University of Technology, Guangzhou, China. His research interests include motion planning, compliant control and computer version.

Liting Zeng will receive the B.A. degree in Fine Arts from the School of Art and Design, Guangdong University of Technology, Guangzhou, China, in 2026. Her research interests include visual design and artistic creation.

Yi Guo received his B.S. degree of Electrical and Computer Engineering from Shanghai Jiao Tong University, Shanghai, China, and his Ph.D. degree of Computer Science and Engineering from Hong Kong University of Science and Technology, Hong Kong, China. He is currently an associate researcher in Shanghai Jiao Tong University. His research interests include embodied AI, collaborative robot, control system. He is a member of the IEEE and the ACM.

Chenguang Yang (Fellow, IEEE) received the B.Eng. degree in measurement and control from Northwestern Polytechnical University, Xian, China, in 2005, and the Ph.D. degree in control engineering from the National University of Singapore, Singapore, in 2010. He performed postdoctoral studies in human robotics at the Imperial College London, London, U.K from 2009 to 2010. Professor Yang holds the Chair in Robotics with Department of Computer Science, University of Liverpool, UK. He is leading the Robotics and Autonomous Systems Group. He was leading Robot Teleoperation Group at Bristol Robotics Laboratory. He is a member of European Academy of Sciences and Arts (EASA). He was awarded UK EPSRC UKRI Innovation Fellowship and individual EU Marie Curie International Incoming Fellowship. He was President of Chinese Automation and Computing Society in the UK. He is the Corresponding Co-Chair of IEEE Technical Committee on Collaborative Automation for Flexible Manufacturing. As the lead author, he won the IEEE Transactions on Robotics Best Paper Award (2012) and IEEE Transactions on Neural Networks and Learning Systems Outstanding Paper Award (2022). His research interest lies in robot control and learning, human robot interaction and intelligent system design.

APPENDIX

A. Prompt Engineering

In our experiment, multiple models were utilized, including LLM and VLM. Before program execution, the LLM, in combination with a skill knowledge library, translates high-level human instructions into machine-interpretable low-level commands, which are stored in the form of a *dot* file. This *dot* file is then used to generate a corresponding behavior graph that controls the robot in task execution. In the experimental scenario, a monitoring camera captures images and transmits them to the VLM. By analyzing the scene images, the VLM evaluates the execution status of the current subtask and dynamically adjusts the node sequence of the behavior graph in real time. When a critical subtask is reached, a key subtask evaluation process—namely, a VQA procedure—is triggered. At this stage, the VLM assesses task completion and controls the execution of recovery nodes accordingly.

1) Behavior Planning Prompt: The following prompt is designed to guide the LLM in correctly generating *dot* files. It includes the fundamental parameters of the robot, basic concepts from the skill knowledge library, the basic structure and skill tags of the skill knowledge library, an example application, and the output format of the *dot* file.

```
[system prompt]
- You function as a robotic controller
    that selects appropriate skills from a
     predefined behavior and vision
    library to accomplish user-specified
    tasks. The output is a dot file that
    constructs the behavior graph for a
    robotic manipulator. The robot is a
    six-degree-of-freedom (6-DOF)
    manipulator equipped with an end-
    effector depth camera for capturing
    depth information of the scene. In
    addition to the manipulator, an
    external global camera is present in
    the environment, capturing images of
    the entire scene to monitor task
    execution status.
[motion library]
- 'Homing': This skill resets the robotic
     arm to its home position, restoring
    it to a predefined reference point for
     the purpose of initiating or
    reinitiating subsequent tasks with
    consistency and accuracy.
- 'Approach': This skill enables the
    robotic arm to move toward the
    designated target location where the
    target object is situated, ensuring
    precise positioning for subsequent
    manipulation tasks.
- 'Grasp': This skill equips the robotic
    arm with the capability to securely
    grasp the designated target object,
    ensuring precise and stable
    manipulation during the execution of
    various tasks.
- 'Spoon': This skill enables the robotic
     arm to effectively scoop up the
    designated target object, facilitating
     precise handling and manipulation in
    various operational scenarios.
- 'Pour': This skill provides the robotic
     arm with the capability to securely
    grasp target objects, ensuring stable
    and precise manipulation for a wide
    range of tasks.
- 'Release': This skill enables the
    robotic arm to accurately release the
    target object, ensuring controlled
    placement and efficient task execution
     in various operational scenarios.
- 'Recovery': This skill enables the
    robotic arm to re-execute the original
     task upon detecting a task failure.
[vision library]
- 'Objection Detection': This skill
    allows the robotic arm to use the
    camera at its end-effector to obtain
    the three-dimensional coordinates of
    the target object.
- 'Object Correction': This skill enables
     the robotic arm to rotate and
    precisely adjust the 3D pose of the
    target object, ensuring accurate
    positioning and alignment for optimal
    task execution.
- 'Failure Detection': This capability
    operates continuously during program
    execution to monitor the scene for
    potential sub-task failures. Upon
    detection of a failure, it returns a
    failure value, enabling robust failure
     detection. When a task fails, the
```

```
   system can re-execute specific sub-
   tasks to ensure task completion.
[example applications]
- Here is an example application. When
   generating the dot file, please follow
    the corresponding format.
- Input: Please spoon sugar into coffee.
- Output:
digraph TaskGraph {
    Home -> "Detect Sugar";
    Home -> "Detect Coffee";

    "Find Sugar" -> "Find Coffee";
    "Find Sugar" -> "Move to Sugar";

    "Find Coffee" -> "Find Sugar";
    "Find Coffee" -> "Move to Coffee";

    "Move to Sugar" -> "Scoop Sugar";
    "Move to Coffee" -> "Put Sugar into
       Coffee";

    "VLM-Based Online Monitoring" -> "
       Home";
    "VLM-Based Online Monitoring" -> "
       Move to Sugar";
    "VLM-Based Online Monitoring" -> "
       Move to Coffee";

    "Failure Detection and Recovery" -> "
       Move to Sugar";
    "Failure Detection and Recovery" -> "
       Move to Coffee";
}
[user prompt]
- This is an image depicting the scene
   prior to the robot initiating the task
    {scene_image}. In the image, both the
    robot and the task props are prepared
   . Please generate a behavior graph to
   control the robot in accomplishing the
    {user_task}, and store it in the
   required format.
```

Prompt 1: Behavior Planning Prompt

2) Environmental Monitoring Prompt: In the following prompt, the VLM integrates the current subtask status and the surrounding environment. When unexpected external disturbances occur, the VLM analyzes the robot's state and the nature of the disturbance, then regenerates the task graph code. It dynamically adjusts the node sequence of the behavior graph to attempt error correction and ensure task success.The previous section regarding the skill knowledge library will also be provided to the VLM.

```
[system prompt]
- You are a robot state inspector, and
   you need to monitor the gap between
   task execution and planning in real
   time. When interference or abnormal
   subtask execution states occur, you
   must stop the robot's operation. Then,
    based on the scene and the robot's
   status, you will need to replan the
   behavior graph, considering the
   logical relationships between
   completed tasks, unfinished tasks, and
    abnormal tasks.
[user prompt]
- The following are some basic exceptions
    and their corresponding solutions:
1. If the target is moved while the robot
    is near it:
Solution: Return to the home node and re-
   detect the target.
2. If an abnormal task state occurs
   during execution:
Solution: Re-execute the node where the
   abnormal state was detected.
```

Prompt 2: Environmental Monitoring Prompt

3) Subtasks Evaluation Prompt: The critical subtask prompt is used to assess task execution when a critical subtask is being performed. By passing in an image and interacting with the VLM, the robotic arm is controlled to either proceed to the next subtask or reattempt the current one. If the VLM responds with "Yes," the robot moves to the next node in the behavior tree. If the VLM responds with "No," a recovery node is triggered, prompting the robot to recall the previous node and re-execute the last sub-behavior API. This triggers a repeated VQA process until the task is successfully completed or the maximum retry limit is reached. During the execution of the behavior graph, each failure detection node returns one of two signals: success (1) or failure (0).

```
[system prompt]
- You are a state inspector. When a
   critical subtask is being executed,
   you need to determine its execution
   status based on the interaction
   between the robot and the scene in the
    input image.
[user prompt]
- This is an image of the robot
   performing its task, specifically
   executing the critical subtask {
   key_subtask}.
- Based on the input image, answer the
   following question:
1. Has the robot successfully executed {
   key_subtask}?
- Without any additional output, return 1
    if the task is successful and 0 if it
    is not.
- Based on the input image and the robot'
   s status, replan the robot's behavior
```

```
graph and save it in the specified
format.
```

Prompt 3: Subtasks Evaluation Prompt