LOGIV: Logic-Graph Inference with VAL-Verification for Long-Horizon Robotic Manipulation

Enqiao Lu [1] **Zhenglin Wan** [2] **Xingrui Yu** [3] **Pengfei Zhou** [2] **Wangbo Zhao** [2] **Yang You** [2] **Ivor Tsang** [3 4]

Abstract

Long-term robotic operation has long been plagued by **temporal failures** during execution, as static task instructions (i.e., language conditions) fail to provide dynamic guidance for complex stages. This results in **stage confusion** and goal deviation in multi-stage scenarios, despite the strong short-term reaction capabilities of the underlying actuators. In this paper, we demonstrate that achieving robust long-term autonomy does not require retraining the actuators or complex hierarchical architectures, but rather **temporal reparameterization of the task-instruction interface**. We propose a **training-free dual-system** reasoning framework, **LOGIV** (LOgic-Graph Inference with VAL-verification), which decomposes global instructions into dynamic, multi-stage natural language priors. To ensure logical consistency, we introduce a graph-based self-correction mechanism that utilizes formal verification and standardized repair operators to autonomously correct plans generated by large language models. Experimental results on datasets such as DROID, AgiBot, EgoDex, and RoboTwin 2.0 show that our framework significantly improves task success rates without modifying the underlying model weights, with more pronounced effects on more complex tasks. Due to its architecture-agnostic and lightweight nature, LOGIV offers an efficient and **plug-and-play** solution for bridging the long-term temporal gap in existing world action models.

[1]The Chinese University of Hong Kong, Shenzhen [2]National University of Singapore [3]CFAR and IHPC, Agency for Science, Technology and Research (A*STAR), Singapore [4]Nanyang Technological University. Correspondence to: Xingrui Yu <Yu_Xingrui@a-star.edu.sg>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

1. Introduction

To address the core challenge of long-horizon robot operation, where task failure stems not from single-step inaccuracies but from the lack of stable stage perception and transition timing, we must move beyond reliable actuators to prevent the stage confusion and goal deviation prevalent in complex multi-stage tasks (Chen et al., 2025; Liu et al., 2026; Dai et al., 2026; Fan et al., 2025; Du et al., 2026; Yan et al., 2026; Zhang et al., 2024).Vision Language Action Models (VLA) and World Action Models (WAM) are the two mainstream technical routes exist in robot operation. VLAs achieve flexible zero-shot generalization in open-vocabulary semantic operations via large-scale language-visual alignment but essentially lean toward short-term action mapping, struggling with multi-stage, long-span temporal logical dependencies (Brohan et al., 2022; Zitkovich et al., 2023; Kim et al., 2024; Bagaria et al., 2026; Huang et al., 2026; Yang et al., 2025). In contrast, WAM provides robots with stronger temporal memory and future state prediction through implicit world modeling and closed-loop prediction, supporting long-term decision-making (Ha & Schmidhuber, 2018; Hafner et al., 2019; 2023). However, existing WAMs have a key defect: limited reasoning frequency and long-term task forgetting. Static global instructions fail to guide the dynamic execution process, leading to lost stage positioning, step disorder and goal drift, even with accurate underlying actions (Ye et al., 2026; Gao et al., 2026; Du et al., 2026; Zhang et al., 2024). Our method is thus more suitable for WAM adaptation, and does not affect the original reasoning frequency even with new modules added.

The current approaches to addressing the challenges of long-horizon tasks mainly fall into three core directions, none of which have tackled the core bottleneck of "**insufficient temporal expressiveness of the task-instruction interface**". The first direction expands and generalizes low-level policies through larger-scale data, more uniform action spaces, and stronger multimodal representations (Brohan et al., 2022; O'Neill et al., 2024; Team et al., 2024; Kim et al., 2024). The core limitation here is an excessive focus on actuator optimization while completely neglecting the temporal dimension design of the task-instruction interface. The

second direction introduces advanced language reasoning, using large language models or visual language models to generate step plans that drive downstream controllers (Ahn et al., 2022; Huang et al., 2022; 2023; Han et al., 2025). The core limitation here is treating the planner and actuator as modules that need to be co-trained, without taking the task-instruction interface as the optimization core, thus requiring modifications to model parameters or architectures. The third direction involves hierarchical reinforcement learning and temporal abstraction methods (Sutton et al., 1999; Nachum et al., 2018; Yang et al., 2026), which, although emphasizing the importance of hierarchical structures, have the core limitation of relying on explicit hierarchical policy training and additional supervision signals, thus requiring modifications to the actuator parameter structure.

In contrast to the three mentioned paradigms,We propose LOGIV, a training-free framework that enhances long-horizon WAM capabilities without weight fine-tuning or human supervision. LOGIV shifts planning from fragile linear sequences to structured Logic-Graphs, where nodes represent atomic instructions and edges encode temporal and causal dependencies. This converts probabilistic plans into verifiable, locally editable symbolic structures. By integrating the VAL formal verification tool (Howey et al., 2004), LOGIV rigorously audits graph integrity to localize logical conflicts like prerequisite omissions. Once detected, deterministic repair operators trigger localized refinement and iterative correction, ensuring coherence without regenerating entire plans. This closed loop of representation, verification, and refinement marks a leap from probabilistic generation to deterministic constraints, overcoming the uncontrollability and short-sightedness of frozen WAMs.

Through systematic evaluations on three types of heterogeneous base models, namely DreamZero (Ye et al., 2026), DreamDojo (Gao et al., 2026), and LingBot-VA (Wu et al., 2026),the experimental results comprehensively verified the superiority of the LOGIV framework. Macro experiments across datasets demonstrated that LOGIV effectively mitigated the sharp performance decline of base models as the task duration increased. By introducing dynamic phase instructions, the model exhibited remarkable robustness in handling long-sequence tasks, successfully overcoming the "long-range disaster". In comparisons within different complexity levels of each dataset, LOGIV achieved significant performance leaps. Not only did it excel in Success Rate, but it also showed stable improvements in fine-grained metrics such as Mean Loss and Task Progress, demonstrating its precise guiding role at the micro-execution level. Additionally, Efficiency Profiling confirmed that LOGIV incurs only a millisecond-level fixed overhead at the startup stage, without interfering with the inference frequency during the execution phase.The specific contributions of this work can be summarized as follows:

- LOGIV introduces a dynamic multi-stage instruction interface, converting static global instructions into stage instructions that evolve with execution progress.
- To address the logical illusion of LLMs, LOGIV designs an integrated self-correction loop combining graph structure and VAL verification to correct instruction logic through a standardized process..
- LOGIV constructs a dual-system freezing architecture, formalizing the long-term bottleneck as a reparameterization problem in the reasoning stage and discarding the parameter update of the executor.
- Extensive experiments on three major WAMs confirm consistent gains of LOGIV. Ablations isolate the value of the two core modules, and efficiency profiling quantifies deployment overhead.

2. Related work

World Action Model (WAM) has become one of the mainstream foundations for robot control via implicit environmental dynamic modeling and future state-action joint prediction (Ye et al., 2026; Gao et al., 2026; Wu et al., 2026). New-generation WAMs like DreamZero (Ye et al., 2026), DreamDojo (Gao et al., 2026), and LingBot-VA (Wu et al., 2026) adopt language-conditioned driving and video-action joint generation, enhancing short-term closed-loop control and single-step execution. Recent works such as GigaWorld-Policy, AIM, DyWA advance architecture and generalization (Ye et al., 2026; Fan et al., 2026; Lyu et al., 2025); WorldVLA, LAC-WM, IRASim expand WAM's boundaries (Cen et al., 2025; Zhu et al., 2025). Despite rapid iterations, modern WAMs face temporal feature collapse, stage awareness lack, and target deviation in long-term tasks, rooted in reliance on static global instructions and lack of evolvable temporal guidance (Du et al., 2026; Zhang et al., 2024; Fan et al., 2025). Existing optimizations focus on model structure and training, ignoring task-instruction interface defects. In contrast, our LOGIV freezes underlying WAM weights, and enables training-free and plug-and-play logical enhancement via inference-stage logic graph reasoning, complementing mainstream WAM paradigms.

High-level planning methods for long-horizon tasks mainly fall into two technical routes: hierarchical decision-making and language-guided planning. Hierarchical reinforcement learning and temporal abstraction decompose complex tasks via multi-level strategies; classic works like Options, MAXQ, Feudal RL, HIRO, HAC abstract long action sequences into reusable units, emphasizing phased decision-making (Sutton et al., 1999; Dayan & Hinton, 1992; Nachum et al., 2018). However, they rely on explicit hierarchical training, artificial supervision, and specific goal representations, requiring executor structure modification and failing to adapt to frozen models (Nachum

et al., 2018; Levy et al., 2017). Another route uses LLMs as planners: SayCan aligns high-level skills with low-level feasibility (Ahn et al., 2022), Inner Monologue introduces environmental feedback (Huang et al., 2022), and Code as Policies, VoxPoser, RT-H convert semantics into executable constraints (Liang et al., 2023; Huang et al., 2023; Belkhale et al., 2024). These methods enhance planning expressiveness but treat planner and executor as a coupled system requiring fine-tuning, with unconstrained step dependencies causing severe logical hallucinations (Ahn et al., 2022; Huang et al., 2022). Unlike them, LOGIV adopts task phase decomposition but abandons training dependencies, uses logic graphs to model step dependencies, achieving training-free, auto-correcting high-level planning.

Automatic Plan Validation. VAL is a representative error-correction work and classic symbolic formal verification tool for robot task planning. Its core is strict logical executability checks on action sequences' preconditions, execution effects, parameter bindings, and temporal dependencies (Howey et al., 2004; Fox & Long, 2003). VAL takes PDDL-style symbolic inputs, simulates execution via forward state inference to judge plan legality, outputting binary judgments, precise error localization, and machine-parsable error types instead of vague feedback. Existing works often combine formal verification with LLM planning to alleviate logical failures (Gaglione et al., 2022; Lee et al., 2025). Some use LLM self-reflection for optimization but rely on probabilistic generation, failing to eliminate temporal and precondition errors (Madaan et al., 2023); others use LTL logic or symbolic rules for correction (Gaglione et al., 2022; Li & Zhao, 2025). Methods like SayCan (Ahn et al., 2022) and Inner Monologue (Huang et al., 2022) use environment feedback but lack strict symbolic constraints (Huang et al., 2022; Belkhale et al., 2024). These methods verify linear sequences without structured dependencies, cannot locally repair deep errors. In addition, most of them require training embedding, which is incompatible with frozen models (Howey et al., 2004; Gaglione et al., 2022). LOGIV first deeply couples VAL with Logic-Graph Inference, transforming LLM linear plans into dependency-modeling logic graphs compatible with VAL, ensuring long-term planning's logical consistency without training or fine-tuning.

3. Methodology

This section presents preliminaries for WAMs (Section 3.1), followed by a detailed introduction of our LOGIV framework (Section 3.2).

3.1. Preliminary

The World-Action Model (WAM) as the robot's low-level actuator has demonstrated strong short-term closed-loop control capabilities (Ye et al., 2026; Gao et al., 2026; Wu et al., 2026). That is, it can complete precise action feedback and execution based on immediate visual observations, the robot's state, and simple task instructions within a short period. However, in long-term complex tasks, WAM still frequently encounters execution failures, which has become the core bottleneck restricting its practical application. The core function of WAM is to simultaneously predict future visual changes and corresponding execution actions under the joint constraints of task language instructions, real-time visual observations, and the robot's own state. Essentially, it is a closed-loop cycle of "perception-prediction-execution".

Taking the advanced world action model DreamZero as an example (Ye et al., 2026), its closed-loop reasoning process can be further decomposed into the model's joint generation of video and actions at each step, that is, simultaneously predicting the visual scenes and corresponding action sequences for a future period; and after each actual action execution, the real environmental visual observations are written back to the cache. Its cache is a visual cache, storing the historical and current visual latent variables encoded by VAE, correcting prediction deviations through real observations to suppress error accumulation and ensure the accuracy of short-term execution. The core reasoning form of DreamZero can be expressed as:

$$\pi\big(\boldsymbol{a}_{t:t+H}, \boldsymbol{o}_{t:t+H} \mid \boldsymbol{o}_{0:t}, g, \boldsymbol{s}_t\big). \tag{1}$$

Here, $\pi(\cdot)$ represents the prediction distribution of the World-Action Model, responsible for jointly modeling and generating actions and observations for a future period. $a_{t:t+H}$ corresponds to the continuous action sequence output by the model in a single inference, often referred to as an action block, where H indicates the time span of the model's forward prediction in one step. $o_{t:t+H}$ represents the predicted future visual observations corresponding to the above action sequence, providing expected environmental change information in closed-loop execution. $o_{0:t}$ records all historical visual observations from the task start to the current moment, corresponding to the visual cache encoded by VAE, storing the environmental spatiotemporal context and providing the core environmental basis for action decision-making; g and s_t respectively represent the global language instruction corresponding to the task and the robot's execution state at the current moment.

Instead of retraining the underlying WAM, LOGIV reframes long-horizon task failure as an interface-level challenge, reparameterizing the static global instruction g into execution-aligned, phased instructions to maintain dynamic task synchronization without modifying the actuator's parameters.

3.2. Logic-Graph Inference with VAL-Verification

As shown in Figure 1, LOGIV constructs a training-free dual-system collaboration framework. System 1 is the

frozen lower-level WAM executor, and System 2 is the higher-level LLM planner responsible for task-instruction re-parameterization. The synergy between System 1 and System 2 in LOGIV results in a 3-phase pipeline: (1) Logic-Graph Construction & Iterative Refinement, (2) Temporal Re-parameterization, and (3) Closed-Loop Execution. We summarize the full pipeline for LOGIV in Algorithm 1 (Appendix A).

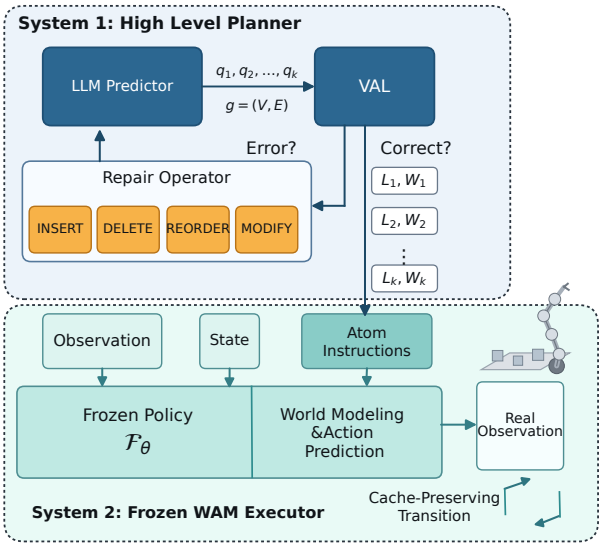

Figure 1. **Overview of the LOGIV framework**. LOGIV adopts a training-free dual-system design. System 1 is the high-level planner: an LLM generates a structured logic graph, which is validated and locally repaired via VAL and repair operators. System 2 is a frozen WAM executor, which receives dynamically stage-aligned instructions to perform closed-loop action prediction, ensuring stable long-horizon execution without modifying the underlying policy.

Phase 1-1: Logic-Graph Construction In the original WAM a single static natural language instruction g is shared across all task stages. It acts as a black-box guidance signal without formal logical structure lacking explicit mappings between preconditions and post-execution effects. It fails to capture logical relevance between task stages especially in long-term planning leading to disjointed coordination ambiguous guidance and inconsistent performance. To mitigate this we leverage an LLM to predict the instruction sequence $\mathcal{L} = [\ell_1, \ell_2, \ldots, \ell_M]$ covering all stages.Each instruction ℓ_i is assigned a relative temporal weight w_i to characterize the semantic duration of the corresponding sub-task. Relying solely on an LLM for atomic instructions is prone to hallucination including omitted predecessor actions incorrect sequences or inconsistent bindings exacerbated by absent logical constraints.

As seen in Figure 2, we define a predicate universe $P = \{p_1, p_2, \ldots, p_n\}$ as the foundational symbolic grounding, where each $p_k \in P$ encapsulates a discrete environmental or task state. To instantiate the predicate universe P, we extract environment-specific object categories and spatial relation states directly from the dataset descriptions. In our offline implementation, the LLM Planner Γ_{LLM} acts as the semantic-to-symbolic translator, parsing the task instructions and visual context into structured PDDL-style literals. While this paper focuses on validating the reasoning structure under a closed predicate universe, deploying LOGIV in completely unconstrained, open-vocabulary environments would ideally require a vision-language grounding module (e.g., an open-vocabulary object detector) to dynamically anchor predicates to real-time visual observations. The structural parameters of $\mathcal{G}$ are formally defined as:

$$v_i = (\ell_i, w_i), \qquad e_{ij} = (\pi_{ij}, \psi_{ij}, \Phi_{ij}), \tag{2}$$

where $v_i \in V$ represents an execution unit with a language command ℓ_i and a semantic weight w_i. Crucially, the edge $e_{ij} \in E$ is not merely a connection but a **logical interface** encapsulating the state-transition dynamics. Here, $\psi_{ij} \subseteq P$ and $\pi_{ij} \subseteq P$ denote the effect and precondition sets, respectively. The parameter Φ_{ij} represents the Action Logic Expression, which specifies the formal nature of the transition. Under this formulation, VAL treats Φ_{ij} as a functional constraint that governs the mapping from the symbolic state to the next action's feasibility. Logical consistency is maintained if the edge satisfies the invariant $\text{Inv}(e_{ij}) : \psi_{ij} \supseteq \pi_{ij}$ under the logic specified by Φ_{ij}.

It is worth noting that ψ_{ij} here does not strictly denote the isolated execution effect of node v_i alone. Rather, within the Logic-Graph framework, ψ_{ij} represents the accumulated forward-propagated symbolic state at edge e_{ij}. This formulation naturally accounts for state persistence, where prerequisites achieved in earlier ancestral nodes or holding in the initial state are preserved across the topology, thereby preventing the constraint from collapsing into naive pairwise edge matching.

Phase 1-2: Iterative Refinement To bridge probabilistic generation with deterministic consistency, we formalize a closed-loop symbolic reasoning paradigm for high-level planning. Specifically, the high-level planning process consists of *Symbolic Verification* and *Local Graph Structure Refinement*, two core components that underpin the main novelty of LOGIV. These components are instantiated in Phase 1 of Algorithm 1, forming the backbone of the logic-graph correction mechanism. We first introduce a symbolic verification oracle $\mathcal{V}$ to rigorously validate the logical consistency of the initial logic-graph $\mathcal{G}^{(0)}$ generated by the LLM planner Γ_{LLM}.

Symbolic Verification. The symbolic verification oracle $\mathcal{V}$ is defined over the graph structural parameters ϕ_i, π_i, ψ_i encoding the intrinsic logical constraints of atomic steps. Formally $\mathcal{V} : \mathcal{G} \times \{\phi_i, \pi_i, \psi_i\} \to \mathcal{E} \cup \{\emptyset\}$ where $\mathcal{E}$ denotes a

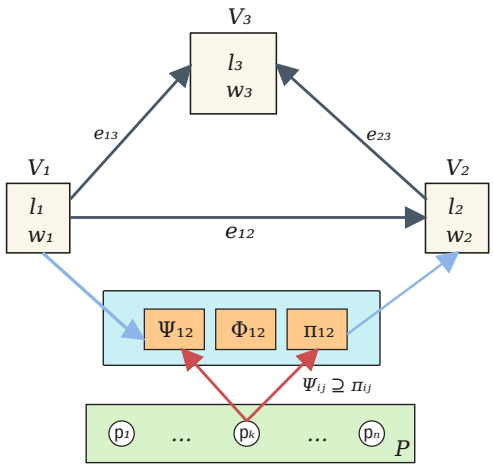

Figure 2. **Formal structure and symbolic grounding of the Logic-Graph.** The logic-graph $\mathcal{G}$ orchestrates atomic action nodes V through functional edge interfaces E, where each edge e_{ij} encapsulates the transition logic Φ_{ij} with symbolic effect set Ψ_{ij} and precondition set Π_{ij}. All logical parameters are grounded in the predicate universe P, enabling localized verification of the causal invariant $\Psi_{ij} \supseteq \Pi_{ij}$ to ensure long-horizon execution consistency.

formal error state consisting of symbolic tuples $\langle v, e, \epsilon \rangle$ with v and e representing errant nodes and edges respectively and ϵ denoting the error type (e.g., missing preconditions, parameter binding conflicts, execution order violations). This formalization enables rigorous symbolic reasoning about the logical integrity of the planning graph, going beyond heuristic error detection to provide a formally grounded verification framework.

Local Graph Structure Refinement. Complementary to the symbolic verification oracle we propose a deterministic repair mapping $\mathcal{R}$ that bridges the gap between symbolic error states and graph refinement operations. $\mathcal{R}$ is formally defined as $\mathcal{R} : \mathcal{E} \rightarrow \mathrm{Op}$, where Op denotes a set of structured graph editing operations $\{\texttt{INSERT}, \texttt{DELETE}, \texttt{REORDER}, \texttt{MODIFY}\}$ each associated with a target node or edge. The mapping $\mathcal{R}$ encodes a set of predefined logical rules (see Appendix B), which are derived from the constraints of long-term planning, ensuring that each repair operation maintains the structural consistency of the graph and avoids the secondary hallucinations inherent in unconstrained text regeneration.

The iterative refinement process leverages Γ_{LLM} to perform local updates on the current graph $\mathcal{G}^{(r)}$ guided by the editing operations from $\mathcal{R}$ rather than regenerating the entire graph. This local refinement strategy is motivated by the principle of minimal intervention, preserving the structural stability of the planning graph while improving correction efficiency. The iterative process is formally characterized as:

$$\mathcal{G}^{(r+1)} = \Gamma_{\text{LLM}}\big(\mathcal{G}^{(r)}, \mathrm{Op}, \mathcal{O}\big), \tag{3}$$

Specifically, we set the hyperparameter r to 0, indicating that we only iterated once. $\mathcal{G}^{(r)}$ denotes the graph structure after the r-th iteration, $\mathcal{O}$ represents fixed environmental observations and task instructions, and Γ_{LLM} denotes the LLM-based refinement operator that optimizes $\mathcal{G}^{(r)}$ to generate $\mathcal{G}^{(r+1)}$. Nodes not subject to verification by $\mathcal{V}$ are excluded from refinement, maintaining the integrity of valid graph components. This entire iterative verification-repair process corresponds to Phase 1 of Algorithm 1 in the appendix, terminating when $\mathcal{V}(\mathcal{G}^{(r)}) = \emptyset$ (logical convergence).

Phase 2: Temporal Re-parameterization Aligning abstract stage-wise instructions with continuous physical execution necessitates a semantic-aware mapping mechanism for temporal re-parameterization. Upon logical convergence, valid atomic instructions and their semantic-driven time weights $\{\ell_i, w_i\}_{i=1}^{M}$ are extracted from $\mathcal{G}^{(r)}$. The adaptive step for each atomic instruction is captured by the time weight w_i, a semantic-aware variable dynamically inferred by Γ_{LLM} based on the inherent complexity and resource requirements of each atomic step. This adaptive inference is a key design choice, as it enables context-aware time allocation across atomic steps without manual intervention. Formally, $w_i = \Gamma_{\text{LLM}}(\ell_i, \mathcal{O})$, capturing the semantic correlation between atomic instructions and their execution demands.

Subsequently, we perform temporal re-parameterization, a methodological step that maps the semantic-driven weights w_i to the total execution horizon T. In essence, this re-parameterization transforms relative semantic weights into absolute physical execution time windows. The normalized weight $\tilde{w}_i$ is defined as:

$$\tilde{w}_i = \frac{w_i}{\sum_{j=1}^{M} w_j}, \tag{4}$$

where $\sum_{j=1}^{M} \tilde{w}_j = 1$ ensuring the weights form a valid probability distribution over the execution progress. We then linearly project $\tilde{w}_i$ to a unique temporal interval $[\tau_{s_i}, \tau_{e_i}]$ for each atomic step ℓ_i:

$$\tau_{s_i} = T \cdot \sum_{j=1}^{i-1} \tilde{w}_j, \qquad \tau_{e_i} = T \cdot \sum_{j=1}^{i} \tilde{w}_j. \tag{5}$$

This projection establishes a direct link between semantic attributes of atomic steps and their temporal execution windows, enabling progress-aligned instruction activation.

Phase 3: Closed-Loop Execution To synchronize the frozen actuator with evolving task stages, we deploy a progress-aligned conditioning interface for real-time interaction. During online execution, the system dynamically maps the current execution time t to the temporal interval $[\tau_{s_i}, \tau_{e_i}]$ activating the synchronized atomic instruction $\ell_t = \ell_i$. This

dynamic activation mechanism is motivated by the need to align task constraints with real-time execution progress, breaking the limitations of static global instructions. The core inference expression for the frozen WAM actuator is formally derived as:

$$\pi\big(a_{t:t+H}, o_{t:t+H} \mid o_{0:t}, \ell_t, s_t\big), \tag{6}$$

where ℓ_t serves as the progress-aligned task constraint replacing static global instructions to provide precise semantic guidance for actuator inference. The frozen configuration of the WAM actuator is enabled by the structured interface provided by the logic-graph, ensuring that high-level planning constraints are seamlessly propagated to the low-level executor without parameter fine-tuning.

Methodologically, the integration of symbolic verification $\mathcal{V}$ and local repair $\mathcal{R}$ establishes a rigorous closed-loop reasoning framework for high-level planning. This framework transforms one-time uncontrollable text generation into a principled process of candidate graph generation, symbolic verification, and local repair of each component grounded in formal definitions. The logic-graph $\mathcal{G}$ serves as a unified interface: enabling normalized phased language conditions for the low-level WAM, providing a formal entry for symbolic verification, and offering a clear space for local refinement. This overall design realizes a training-free, pluggable, and self-correcting dual-system long-term planning paradigm without modifying the underlying WAM parameters.

4. Experiments

Our empirical evaluation is designed to comprehensively assess the feasibility and superiority of the LOGIV framework. Specifically, we aim to answer the following two core research questions:

- **Q1 (Long-Horizon Stability):** Can LOGIV effectively mitigate the "long-horizon disaster" characterized by the severe degradation in success rates among baseline models as task difficulty increases, thereby maintaining robust performance across diverse datasets?
- **Q2 (Fine-Grained Improvement):** Moving beyond macro-level trends, does LOGIV consistently surpass the original base models across distinct complexity splits within each individual dataset?

4.1. Experimental Setup

Base Models and Datasets. This study evaluates LOGIV across three representative world action models including **DreamZero** (Ye et al., 2026), **DreamDojo** (Gao et al., 2026), and **LingBot-VA** (Wu et al., 2026). To ensure fairness and reproducibility, we strictly follow the official protocols of these models and conduct evaluations on their originally designated datasets to maintain consistency with their pre-trained distributions. Specifically, DreamZero is evaluated on **DROID** (Khazatsky et al., 2024) and **AgiBot World** (Bu et al., 2025) using ChatGPT 4o-mini as the high-level planner. DreamDojo is tested on **AgiBot World** and **EgoDex** (Hoque et al., 2025) with ChatGPT 4o-mini, while LingBot-VA is assessed on **RoboTwin 2.0** (Chen et al., 2025) using ChatGPT 5.4. We refrain from evaluating all backbones on all four datasets because cross-dataset testing without domain adaptation would disrupt the training-test alignment and lead to unreliable conclusions.

Dataset Descriptions. The datasets utilized in this study comprehensively cover diverse robot configurations and interaction modes, providing a robust validation ground for long-horizon autonomous manipulation. **DROID** (Khazatsky et al., 2024) is a large-scale, open-source dataset focused on heterogeneous robot manipulation across various real-world environments, featuring diverse camera views and robot states. **AgiBot World** (Bu et al., 2025) provides a comprehensive benchmark for long-range mobile manipulation tasks, specifically designed for testing general-purpose robotic autonomy in unstructured settings. **EgoDex** (Hoque et al., 2025) specializes in first-person perspective dexterous manipulation, emphasizing fine-grained hand-object interactions and complex finger gaiting. **RoboTwin 2.0** (Chen et al., 2025) serves as a high-fidelity dual-arm simulation benchmark designed for evaluating downstream operation standards in complex assembly or sorting tasks.

For all mentioned datasets, we implement a rigorous filtering pipeline to maintain data quality, only retaining episodes that contain successful execution runs, complete multimodal sensor logs, and a trajectory length of at least 10 steps.

Tasks. To rigorously assess the framework's ability to mitigate performance degradation over time, we categorize tasks into three complexity levels designated as L1, L2, and L3. Our motivation for this stratification is to isolate the impact of temporal span on execution stability, as longer horizons typically exacerbate stage confusion and cumulative errors in frozen WAMs. We use the trajectory execution horizon H as a proxy for complexity and divide candidate trajectories based on the 33rd and 66th percentiles of H within each dataset. We implement a deterministic sampling strategy with a fixed seed to select 50 trajectories for each level, employing a round-robin approach across semantic groups to ensure task diversity and prevent high-frequency task bias.

Task Partitioning Details. The categorization of L1, L2, and L3 levels is strictly grounded in the temporal distribution of the underlying trajectory data. Within each dataset, we calculate the cumulative distribution of trajectory lengths H. Trajectories falling within the $[0, 33\%]$ percentile are classified as L1, while those in the $(33\%, 66\%]$ and $(66\%, 100\%]$ percentiles are designated as L2 and L3, respectively. This statistical approach ensures that L3 consistently represents

the most challenging long-horizon scenarios for each specific benchmark. Following this partition, we apply a round-robin sampling strategy to select 50 episodes per level, ensuring that diverse task semantics are represented and preventing the evaluation from being biased by high-frequency repetitive motions.

Temporal Span Metrics. To characterize the physical duration corresponding to these complexity levels, we compute the **Mean Elapsed Time** for each split. For trajectories containing native timestamps, the duration is defined as:

$$T = t_{N-1} - t_0 \tag{7}$$

For data without explicit step-by-step timestamps but possessing a stable sampling frequency f, we utilize the following proxy for duration:

$$T = \frac{H-1}{f} \tag{8}$$

The resulting temporal scaling is summarized in Table 1, illustrating the significant increase in task duration from L1 to L3 across all evaluated benchmarks.

Table 1. Mean elapsed time (in seconds) for each complexity split across datasets.

Dataset	L1 (s)	L2 (s)	L3 (s)
DROID	7.65	13.56	25.05
AgiBot	4.57	6.55	11.00
EgoDex	3.71	12.22	37.47
RoboTwin 2.0	2.44	3.49	6.48

Evaluation Metrics. The main experiment adopts three complementary metrics. Success Rate measures whether the episode-level task is completed. Mean Action Error calculates the average L2 Euclidean distance between the predicted action and the ground-truth action at the alignment sampling moment, reflecting low-level action deviation. Task Progress is a continuous score given based on the initial frame, the true end frame, and the model end frame, used to describe the proximity to the target state. In addition, we report ActionErr < 0.1, the proportion of samples whose action error is below 0.1, to reflect stepwise action alignment accuracy.

4.2. Main Results

In this section, we evaluate the effectiveness of LOGIV by comparing the unmodified original base models, denoted as **Base**, against our complete system, referred to as **Ours**. The comparative performance across three heterogeneous backbone architectures is visually summarized in Figure 3, Figure 4, and Figure 5. These charts collectively form the core evidence for validating our research questions, especially in dealing with the performance fluctuations that arise as task complexity increases.

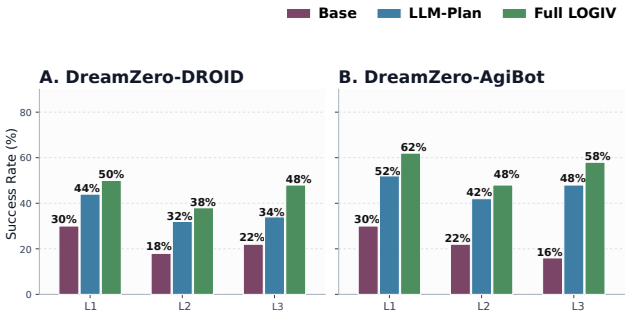

Figure 3. Ablation study on success rate across different complexity splits on dreamzero.

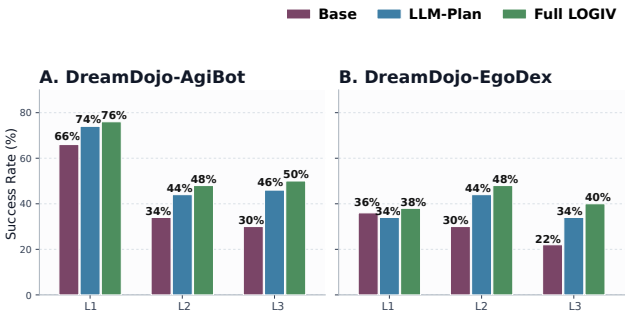

Figure 4. Ablation study on success rate across different complexity splits on dreamdojo.

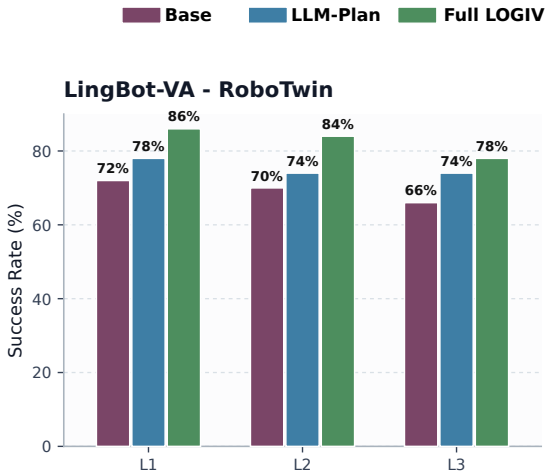

Figure 5. Ablation study on success rate across different complexity splits on Lingbot-VA.

Addressing **Q1** regarding long-horizon stability, the results presented in Figure 3, Figure 4, and Figure 5 consistently reveal a significant disparity in robustness between the two systems. As task difficulty scales from L1 to L3, the **Base** models exhibit a characteristic performance collapse where success rates drop precipitously due to cumulative temporal errors and stage confusion. In stark contrast, our proposed LOGIV framework maintains a much flatter decay curve across all three benchmarks and sustains high success rates even in the most challenging L3 scenarios. This uniform

improvement across diverse backbones demonstrates that by re-parameterizing the task-instruction interface, LOGIV successfully mitigates the long-horizon disaster that typically plagues frozen WAMs.

Regarding **Q2** and the pursuit of fine-grained gains, the substantial gaps between the **Base** and **Ours** bars across all complexity splits confirm that the benefits of LOGIV are substantive at every operational stage. Beyond the macro-level success rates shown in the figures, the framework improves micro-level metrics including Mean Loss and Task Progress by providing precise and stage-aligned semantic guidance. This guidance prevents the executor from entering irreversible failure states and ensures that the final state remains in close proximity to the intended goal. These comprehensive improvements validate that upgrading the high-level reasoning interface is a highly effective and plug-and-play paradigm for complex manipulation. The complete quantitative results of the detailed assessment indicators can be obtained from the Appendix C.

An interesting cross-level trend in Table 2 is that the performance gain (Δ) is highly pronounced in L3 (long-horizon) scenarios but relatively moderate in L2 splits. We attribute this to the baseline model's intrinsic error-tolerance threshold. In L1/L2 tasks, the frozen WAM can occasionally rely on its short-term closed-loop visual reactive capabilities to correct minor drift. However, as the task transitions to L3, the accumulated temporal forgetting triggers a catastrophic breakdown, where LOGIV's structured Logic-Graph serves as a critical deterministic scaffold. For mid-level (L2) tasks, the linear projection of semantic weights faces a tighter marginal error window, making them more sensitive to boundary misalignments than L3 tasks where structural guiding dominates.

4.3. Ablation Studies

To decouple the contributions of each component in the dual-system framework, we conducted an ablation study on dynamic conditional reparameterization and validation-driven error correction mechanisms.

The effect of dynamic task re-parameterization. First, we explore the impact of instruction expression on long-horizon tasks. We denote the version that only includes LLM stage splitting but removes the self-correction mechanism as **LLM-Plan**. As shown in Figures 3, 4, and 5, compared with the Base model, LLM-Plan demonstrates a significant increase in success rates across all task levels.

This result reveals that the essence of long-term execution failure is the feature collapse of static task conditions in the temporal dimension. LLM-Plan performs reparameterization by mapping the semantic information of long-term goals into local atomic constraints that evolve with execution progress p_t. Even without logical calibration, this dynamic interface significantly reduces the search space of the underlying WAM during closed-loop generation.

Suppression of logical hallucination via deterministic repair. Furthermore, we compared the logical correctness of three high-level planning strategies to verify the necessity of the VAL mechanism. For this experiment, we did not perform model reasoning but directly used different error correction methods to correct each dataset separately. We compared three scenarios: no error correction, using LLM for self-correction, and using the VAL model for error correction, which are represented as **Raw Plan**, **LLM Self-Refine**, and **VAL-Corrected** in the table below. The final results are shown in Table 6. We found that although LLM Self-Refine outperformed the raw plan, it still led to about 26% of missing preconditions due to logical hallucinations when dealing with complex long sequences like DROID. Our VAL error correction method can perfectly solve the self-correction hallucination problem of LLM.

The comparison in Table 6 shows that robustness of long-horizon planning should not rely on the sampling probability of the LLM, but should be established on a deterministic closed loop of generation–verification–correction.

5. Conclusion

In summary, we propose LOGIV, a dual-system framework for long-horizon robotic operations. LOGIV formalizes long-horizon tasks as a reasoning reparameterization process, decomposing complex goals into a structured Logic-Graph to eliminate decision-making drift. Central to our approach is a symbolic self-correction module that monitors execution and performs localized structural repairs, effectively mitigating error accumulation and execution instability. Extensive evaluations on representative benchmarks demonstrate that LOGIV significantly outperforms mainstream methods in task success rates and robustness. This work provides a reliable, training-free paradigm for complex robotic planning, advancing the practical deployment of general robot systems in real-world scenarios.

Limitation: Currently, LOGIV maps semantic weights to physical time windows linearly. While this training-free interface yields significant gains across multiple WAM benchmarks, it introduces a structural limitation: the high-level planner operates open-loop relative to low-level temporal anomalies (e.g., execution delays or premature completions). If an actuator fails to achieve a sub-goal within its allocated window $[\tau_{s_i}, \tau_{e_i}]$, the system blindly advances. Future work will explore integrating an asynchronous event-triggered signaling mechanism, allowing the underlying WAM to feedback execution completion cues to dynamically update the interval boundaries of the Logic-Graph.

References

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Bagaria, V., Sebastian, B., and Patel, N. K. Recursive belief vision language action models. *arXiv preprint arXiv:2602.20659*, 2026.

Belkhale, S., Ding, T., Xiao, T., Sermanet, P., Vuong, Q., Tompson, J., Chebotar, Y., Dwibedi, D., and Sadigh, D. Rt-h: Action hierarchies using language. *arXiv preprint arXiv:2403.01823*, 2024.

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

Bu, Q., Cai, J., Chen, L., Cui, X., Ding, Y., Feng, S., Gao, S., He, X., Hu, X., Huang, X., et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

Cen, J., Yu, C., Yuan, H., Jiang, Y., Huang, S., Guo, J., Li, X., Song, Y., Luo, H., Wang, F., et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

Chen, T., Chen, Z., Chen, B., Cai, Z., Liu, Y., Li, Z., Liang, Q., Lin, X., Ge, Y., Gu, Z., et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Dai, T., Han, M., Du, T., Liu, Z., Li, Z., Khan, S., Yu, J., and Chang, X. See, plan, rewind: Progress-aware vision-language-action models for robust robotic manipulation. *arXiv preprint arXiv:2603.09292*, 2026.

Dayan, P. and Hinton, G. E. Feudal reinforcement learning. *Advances in neural information processing systems*, 5, 1992.

Du, X., Qian, J., Zhou, S., and Schoellig, A. P. Perceptive hierarchical-task mpc for sequential mobile manipulation in unstructured semi-static environments. *arXiv preprint arXiv:2603.10227*, 2026.

Fan, L., Xu, Z., Cao, C., Zhang, W., Yuan, M., and Chen, J. Aim: Intent-aware unified world action modeling with spatial value maps. *arXiv preprint arXiv:2604.11135*, 2026.

Fan, Y., Ding, P., Bai, S., Tong, X., Zhu, Y., Lu, H., Dai, F., Zhao, W., Liu, Y., Huang, S., et al. Long-vla: Unleashing long-horizon capability of vision language action model for robot manipulation. *arXiv preprint arXiv:2508.19958*, 2025.

Fox, M. and Long, D. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*, 20:61–124, 2003.

Gaglione, J.-R., Neider, D., Roy, R., Topcu, U., and Xu, Z. Maxsat-based temporal logic inference from noisy data: J.-r. gaglione et al. *Innovations in Systems and Software Engineering*, 18(3):427–442, 2022.

Gao, S., Liang, W., Zheng, K., Malik, A., Ye, S., Yu, S., Tseng, W.-C., Dong, Y., Mo, K., Lin, C.-H., et al. Dreamdojo: A generalist robot world model from large-scale human videos. *arXiv preprint arXiv:2602.06949*, 2026.

Ha, D. and Schmidhuber, J. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Han, S., Qiu, B., Liao, Y., Huang, S., Gao, C., Yan, S., and Liu, S. Robocerebra: A large-scale benchmark for long-horizon robotic manipulation evaluation. *arXiv preprint arXiv:2506.06677*, 2025.

Hoque, R., Huang, P., Yoon, D. J., Sivapurapu, M., and Zhang, J. Egodex: Learning dexterous manipulation from large-scale egocentric video. *arXiv preprint arXiv:2505.11709*, 2025.

Howey, R., Long, D., and Fox, M. Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pp. 294–301. IEEE, 2004.

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

Huang, W., Wang, C., Zhang, R., Li, Y., Wu, J., and Fei-Fei, L. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

Huang, Y., Wu, J., Bu, W., Xiong, Z., Jiang, G., Li, Y., Ji, K., Xie, S., Huang, Y., Wu, C., et al. Robostream: Weaving spatio-temporal reasoning with memory in vision-language models for robotics. *arXiv preprint arXiv:2603.12939*, 2026.

Khazatsky, A., Pertsch, K., Nair, S., Balakrishna, A., Dasari, S., Karamcheti, S., Nasiriany, S., Srirama, M. K., Chen, L. Y., Ellis, K., et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Lam, G., Sanketi, P., et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Lee, C. P., Porfirio, D., Wang, X. J., Zhao, K. C., and Mutlu, B. Veriplan: Integrating formal verification and llms into end-user planning. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 1–19, 2025.

Levy, A., Konidaris, G., Platt, R., and Saenko, K. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017.

Li, J. and Zhao, G. T3 planner: A self-correcting llm framework for robotic motion planning with temporal logic. *arXiv preprint arXiv:2510.16767*, 2025.

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

Liu, Y., Zhu, J., Mo, Y., Li, G., Cao, X., Jin, J., Shen, Y., Li, Z., Yu, T., Yuan, W., et al. Palm: Progress-aware policy learning via affordance reasoning for long-horizon robotic manipulation. *arXiv preprint arXiv:2601.07060*, 2026.

Lyu, J., Li, Z., Shi, X., Xu, C., Wang, Y., and Wang, H. Dywa: Dynamics-adaptive world action model for generalizable non-prehensile manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11058–11068, 2025.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

O'Neill, A., Rehman, A., Maddukuri, A., Gupta, A., Padalkar, A., Lee, A., Pooley, A., Gupta, A., Mandlekar, A., Jain, A., et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999.

Team, O. M., Ghosh, D., Walke, H., Pertsch, K., Black, K., Mees, O., Dasari, S., Hejna, J., Kreiman, T., Xu, C., et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Wu, W., Lu, F., Wang, Y., Yang, S., Liu, S., Wang, F., Zhu, Q., Sun, H., Wang, Y., Ma, S., et al. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*, 2026.

Yan, H., Li, Q., Yang, J., and Mu, Y. Progressvla: Progress-guided diffusion policy for vision-language robotic manipulation. *arXiv preprint arXiv:2603.27670*, 2026.

Yang, J., Zhu, B., Chen, J., and Jiang, Y.-G. Actor-critic for continuous action chunks: A reinforcement learning framework for long-horizon robotic manipulation with sparse reward. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18692–18700, 2026.

Yang, Y., Sun, J., Kou, S., Wang, Y., and Deng, Z. Lohovla: A unified vision-language-action model for long-horizon embodied tasks, 2025. URL `https://arxiv.org/abs/2506.00411`.

Ye, S., Ge, Y., Zheng, K., Gao, S., Yu, S., Kurian, G., Indupuru, S., Tan, Y. L., Zhu, C., Xiang, J., et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Zhang, Y., Xue, T., Razmjoo, A., and Calinon, S. Logic dynamic movement primitives for long-horizon manipulation tasks in dynamic environments. *CoRR*, 2024.

Zhu, F., Wu, H., Guo, S., Liu, Y., Cheang, C., and Kong, T. Irasim: A fine-grained world model for robot manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9834–9844, 2025.

Zitkovich, B., Yu, T., Xu, S., Xu, P., Xiao, T., Xia, F., Wu, J., Wohlhart, P., Welker, S., Wahid, A., et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

A. Algorithm

Algorithm 1 LOGIV Framework Pipeline

Input: Initial Observation o_0, Initial State s_0, Global Task Prompt g, Total Execution Horizon T
Models: LLM Planner Γ_{LLM}, Verifier VAL, Repair Operator $\mathcal{R}$, Frozen Policy π_θ
Output: Executed Action Trajectory

 # Phase 1: Logic-Graph Construction & Iterative Refinement (Offline)
Initialize iteration $r \leftarrow 0$.
Generate initial logic graph $\mathcal{G}^{(0)} \leftarrow \Gamma_{\text{LLM}}(g, o_0)$.
$\mathcal{E} \leftarrow \text{VAL}(\mathcal{G}^{(0)})$ *# Extract formal error state*
while $\mathcal{E} \neq \emptyset$ **do**
  $Op \leftarrow \mathcal{R}(\mathcal{E})$ *# Map errors to $Op \in \{$INSERT, DELETE, REORDER, MODIFY$\}$*
  $\mathcal{G}^{(r+1)} \leftarrow \Gamma_{\text{LLM}}(\mathcal{G}^{(r)}, Op, o_0)$ *# Local graph refinement*
  $\mathcal{E} \leftarrow \text{VAL}(\mathcal{G}^{(r+1)})$ *# Iterative refinement*
  $r \leftarrow r + 1$
end while
Extract valid instructions and weights $\{\ell_i, w_i\}_{i=1}^{M}$ from graph $\mathcal{G}^{(r)}$.

 # Phase 2: Temporal Re-parameterization (Offline)
for $i = 1$ **to** M **do**
  $\tilde{w}_i \leftarrow \frac{w_i}{\sum_{j=1}^{M} w_j}$ *# Weight normalization*
  $[\tau_i^s, \tau_i^e] \leftarrow \left[T \sum_{j=1}^{i-1} \tilde{w}_j,\ T \sum_{j=1}^{i} \tilde{w}_j\right]$ *# Temporal projection*
end for

 # Phase 3: Closed-Loop Execution (Online)
$t \leftarrow 0$, $o_{0:t} \leftarrow o_0$
while $t < T$ **do**
  Identify current stage index i such that $t \in [\tau_i^s, \tau_i^e]$.
  $\ell_t \leftarrow \ell_i$ *# Activate stage-aligned instruction*
  $a_{t:t+H}, o_{t:t+H} \leftarrow \pi_\theta(\cdot \mid o_{0:t}, \ell_t, s_t)$ *# Frozen WAM inference*
  Execute $a_{t:t+H}$, update observation history $o_{0:t+H}$ and state s_{t+H}.
  $t \leftarrow t + H$
end while

B. Formal Definitions of Predefined Logical Rules $\mathcal{R}$

In the LOGIV framework, the set of predefined logical rules R serves as a family of constraint functions acting on the logic-graph $\mathcal{G} = (V, E)$. The repair mapping $\mathcal{R}$ classifies error states $\mathcal{E}$ based on these rules and triggers corresponding graph editing operations Op. The formal definitions and their implications are detailed below:

Causal Consistency and State Transition Invariants. The fundamental prerequisite for long-term plan feasibility is the causal alignment across task stages. For any directed edge $e_{ij} \in E$ connecting source node v_i to target node v_j, the logical parameters must satisfy the subset inclusion invariant:

$$\text{Rule}_{\text{causal}} : \forall e_{ij} \in E, \quad \psi_{ij} \supseteq \pi_{ij} \tag{9}$$

where $\psi_{ij} \subseteq P$ represents the effect set generated by v_i and $\pi_{ij} \subseteq P$ denotes the precondition set required by v_j, both encapsulated within the edge e_{ij}. A violation, defined as $\psi_{ij} \not\supseteq \pi_{ij}$, is categorized as a *Hallucinated Dependency*. This rule forces the LLM to transition from probabilistic association to deterministic grounding via the action logic expression Φ_{ij}. While strict inclusion eliminates execution-level hallucinations, it may introduce an exploration-exploitation trade-off where the system becomes hypersensitive to minor semantic variations, potentially leading to over-repairing in open-vocabulary environments.

Topological Integrity and Structural Acyclicity. To prevent the system from entering infinite logical loops or encountering isolated sub-tasks, the logic-graph must maintain structural closure and acyclicity within the symbolic space:

$$\text{Rule}_{\text{topo}} : \nexists\{v_1, v_2, \ldots, v_k\} \subseteq V \text{ s.t. } (v_k, v_1) \in E \wedge \text{Path}(v_1, v_k) \tag{10}$$

Formally, $\mathcal{G}$ must satisfy the properties of a Directed Acyclic Graph (DAG). If a cycle is detected across edges $\{e_{ij}\}$, the repair mapping $\mathcal{R}$ invokes `DELETE` or `REORDER` operators to resolve the conflict. This structural constraint ensures that the high-level plan possesses a clear directed causality, which is essential for the frozen System 1 executor to maintain temporal progress. By pruning cyclic hallucinations, this rule guarantees that the planning horizon is finite and execution-efficient, avoiding the "reasoning traps" common in unconstrained LLM decoders.

Temporal Monotonicity and Boundary Projection. The alignment of abstract semantic instructions with physical time windows is governed by the monotonicity of time weights w_i. To ensure rational resource allocation and avoid execution collisions, the cumulative duration must respect the global horizon T_{max}:

$$\text{Rule}_{\text{temp}} : \sum_{k=1}^{M} w_k \cdot \text{unit}_T \leq T_{\text{max}} \quad \text{and} \quad \tau_{s_i} < \tau_{e_i} \leq \tau_{s_j} \text{ for } e_{ij} \in E \tag{11}$$

where $[\tau_{s_i}, \tau_{e_i}]$ denotes the absolute temporal window projected from the semantic weights of v_i. This rule constrains the LLM when performing `MODIFY` operations, preventing the generation of overlapping or out-of-boundary execution windows. Theoretically, this establishes a bridge between high-level symbolic reasoning and low-level continuous control. By enforcing temporal monotonicity along edges e_{ij}, LOGIV ensures that the dual-system collaboration remains synchronized with the real-time observation flow $\mathcal{O}$, preventing the actuator from executing stale or premature instructions.

C. Detailed Results

This appendix provides the detailed experimental tables referenced in the main text, which report the main-experiment results across different backbones, benchmarks, and complexity levels. What's more,it shows the performance of different ways of refinement.

Table 2 presents the success rates of the unmodified base models (Base) and our complete LOGIV system (Ours) across different complexity levels (L1, L2, L3) and backbone–benchmark combinations, including the overall average (OA) and the improvement range (ΔOA).

Table 2. Comparison of success rates in the main experiment (%).

Backbone	Benchmark	L1 Base	L1 **Ours**	L2 Base	L2 **Ours**	L3 Base	L3 **Ours**	OA Base	OA **Ours**	ΔOA
DreamZero	DROID	30.0	**50.0**	18.0	**38.0**	22.0	**48.0**	23.3	**45.3**	+22.0
DreamZero	AgiBot	30.0	**62.0**	22.0	**48.0**	16.0	**58.0**	22.7	**56.0**	+33.3
DreamDojo	AgiBot	66.0	**76.0**	34.0	**48.0**	30.0	**50.0**	43.3	**58.0**	+14.7
DreamDojo	EgoDex	36.0	**38.0**	30.0	**48.0**	22.0	**40.0**	29.3	**42.0**	+12.7
LingBot-VA	RoboTwin 2.0	72.0	**86.0**	70.0	**84.0**	66.0	**78.0**	69.3	**82.7**	+13.3

Table 3 reports the Mean Action Error values of Base and Ours in all backbone–benchmark combinations, reflecting the geometric deviation between predicted and ground-truth actions, as well as the reduction in Mean Action Error achieved by LOGIV.

Table 3. Mean Action Error comparison of the main experiment.

Backbone	Benchmark	L1 Base	L1 **Ours**	L2 Base	L2 **Ours**	L3 Base	L3 **Ours**	OA Base	OA **Ours**	ΔOA
DreamZero	DROID	0.1291	**0.1075**	0.1255	**0.1032**	0.1389	**0.1071**	0.1312	**0.1059**	-0.0253
DreamZero	AgiBot	0.0845	**0.0672**	0.0942	**0.0704**	0.0748	**0.0593**	0.0845	**0.0656**	-0.0189
DreamDojo	AgiBot	0.1149	**0.1103**	0.1827	**0.1602**	0.1341	**0.1012**	0.1439	**0.1239**	-0.0200
DreamDojo	EgoDex	0.0974	**0.0792**	0.1057	**0.0810**	0.0987	**0.0820**	0.1006	**0.0807**	-0.0199
LingBot-VA	RoboTwin 2.0	0.6604	**0.6285**	0.6518	**0.6234**	0.6386	**0.6127**	0.6503	**0.6215**	-0.0288

Table 4. Comparison of task progress in the main experiment.

Backbone	Benchmark	L1 Base	L1 **Ours**	L2 Base	L2 **Ours**	L3 Base	L3 **Ours**	OA Base	OA **Ours**	ΔOA
DreamZero	DROID	0.4920	**0.5740**	0.4200	**0.5240**	0.4340	**0.5380**	0.4487	**0.5453**	+0.0966
DreamZero	AgiBot	0.4600	**0.6240**	0.4160	**0.5420**	0.4060	**0.5840**	0.4273	**0.5833**	+0.1560
DreamDojo	AgiBot	0.6080	**0.6820**	0.3820	**0.5720**	0.3100	**0.4560**	0.4333	**0.5700**	+0.1367
DreamDojo	EgoDex	0.4460	**0.4720**	0.4120	**0.5960**	0.4320	**0.5920**	0.4300	**0.5533**	+0.1233
LingBot-VA	RoboTwin 2.0	0.6858	**0.7960**	0.6818	**0.7898**	0.6546	**0.7298**	0.6741	**0.7719**	+0.0978

Table 4 shows the Task Progress scores of Base and Ours at each complexity level, which indicate the proximity of the execution result to the target end state.

Table 5 provides the proportion of samples with action-space error below the threshold of 0.1 (ActionErr < 0.1) for Base and Ours, reflecting the accuracy of stepwise action alignment.

Table 5. Comparison of threshold accuracy proportion ActionErr < 0.1 in the main experiment (%).

Backbone	Benchmark	L1 Base	L1 **Ours**	L2 Base	L2 **Ours**	L3 Base	L3 **Ours**	OA Base	OA **Ours**	ΔOA
DreamZero	DROID	48.7	**66.0**	52.0	**62.7**	48.0	**58.0**	49.6	**62.2**	+12.7
DreamZero	AgiBot	72.0	**89.3**	52.0	**80.0**	77.3	**96.7**	67.1	**88.7**	+21.6
DreamDojo	AgiBot	34.0	**39.0**	24.0	**26.0**	32.0	**38.0**	30.0	**34.3**	+4.3
DreamDojo	EgoDex	65.0	**86.0**	61.0	**71.0**	62.0	**70.0**	62.7	**75.7**	+13.0
LingBot-VA	RoboTwin 2.0	14.8	**15.1**	14.7	**14.5**	19.4	**20.1**	17.1	**17.5**	+0.4

Table 6 shows that the robustness of long-horizon planning should not rely on the sampling probability of the LLM, but should be established on a deterministic closed loop of generation–verification–correction.

Table 6. Logical correctness comparison of three planning strategies.

Dataset	Raw	LLM Self-Refine	**VAL Corrected**
DROID	38.33%	74.00%	**99.67%**
AgiBot	67.50%	87.67%	**100.00%**
EgoDex	92.00%	95.60%	**99.10%**
RoboTwin 2.0	69.33%	75.33%	**98.67%**

Table 7. Comparison of inference time consumption at different complexity levels for each dataset.

Dataset	Split	n	K	Total (s)	T_{plan} (s)
DreamZero–DROID	L1	50	5	10.66	2.308
DreamZero–DROID	L2	50	7	14.93	2.360
DreamZero–DROID	L3	50	10	21.33	2.466
DreamZero–AgiBot	L1	50	24	52.25	3.789
DreamZero–AgiBot	L2	50	33	71.84	3.577
DreamZero–AgiBot	L3	50	55	119.74	3.507

All data in these tables are consistent with the experimental setup and result analysis in the main text, providing detailed support for the conclusions drawn from the main experiments.

D. Analysis of Inference Overhead

In addition to performance metrics, we evaluated time cost. The high-level planning in LOGIV is a one-time decomposition before the start of each trajectory, rather than repeated waiting during execution. The time cost can be divided into start-up phase delay and closed-loop execution phase delay.

Let K denote the actual number of queries made by the underlying policy during real closed-loop execution. For a trajectory of length N, if each policy call on average covers h steps, then $K \approx \lceil N/h \rceil$. The total inference time is approximated as

$$T_{\text{total}} \approx K \cdot T_{\text{policy}} + T_{\text{plan}}. \tag{12}$$

Here, T_{plan} is the one-time planning time before the trajectory starts and $K \cdot T_{\text{policy}}$ is the closed-loop execution time.

Table 7 shows that the planning time of LOGIV remains relatively stable (2.3–3.8 seconds), while total inference time increases with the growth of K. This indicates that as tasks lengthen, the closed-loop execution phase dominates, and the planning time manifests as a fixed start-up delay.