AI

MDPI

Article

A Neuro-Symbolic Framework for LLM-Driven Task Planning and Execution in Industrial Assembly

Dario Antonelli [1,*], Xiaolang Yang [2], Leonardo Urbani [1], Xuemei Liu [2] and Bo Yang [3]

[1] Department of Management and Production Engineering, Politecnico di Torino, 10129 Torino, Italy; s344982@studenti.polito.it
[2] School of Mechanical Engineering, Tongji University, Shanghai 201804, China; yyyxl@tongji.edu.cn (X.Y.); liuxuemei@tongji.edu.cn (X.L.)
[3] Laboratory of Mechanical Transmission, Chongqing University, Chongqing 400044, China; yangbo61@cqu.edu.cn
* Correspondence: dario.antonelli@polito.it

Abstract

This paper proposes a neuro-symbolic framework that combines large language model (LLM)-driven task planning with formal symbolic verification and reactive Behavior Tree execution to support high-precision operations in industrial assembly. While Large Language Models (LLMs) demonstrate expertise in interpreting natural language instructions, they often struggle with processing structured manufacturing data and lack long-horizon planning capabilities and deterministic physical awareness required for complex robotic execution. Furthermore, their susceptibility to generating physically unfeasible sequences and their lack of robust error-recovery mechanisms limit their autonomous deployment. To address these limitations, the proposed pipeline decomposes the assembly process into sequential stages of planning, technological grounding, adaptive execution, and recovery. Initially, the framework leverages LLMs to synthesize multi-modal inputs—ranging from natural language directives to formalized product and process specifications—into a formal Hierarchical Task Network (HTN). To ensure physical feasibility, a deterministic rule-based compiler validates the HTN's syntactic and structural consistency, rejecting undefined symbols and non-decomposable tasks, against established spatial and operational constraints. This verified logical sequence is subsequently compiled into a reactive Behavior Tree (BT) that guides execution through a fixed robotic skill library. During execution, a real-time object detection system and an atomic skill library dynamically translate the BT into grounded actions. Designed for human–robot collaborative environments, the architecture autonomously allocates standard operations to the robotic agent while safely deferring tasks requiring high dexterity to human operators. By combining the semantic adaptability of LLMs with the rigorous state validation of deterministic rule-based tools, this methodology is intended to reduce hallucination-driven failures and support safer manufacturing workflows, as evaluated through a system-level feasibility study rather than a component-wise ablation. The proposed architecture is evaluated against both physical and logical criteria using task-specific evaluation metrics. The framework was evaluated on four assembly tasks (T1–T4) using two Gemma model configurations, with 30 trials per task for each configuration (120 trials per configuration; 240 trials in total), achieving near-100% plan validity and near-zero hallucination rates under the higher-capacity configuration. Future integration of Vision-Language-Action (VLA) policies at the level of individual Behavior Tree leaves is discussed in Section Perception and Dynamic Adaptation as a promising direction to replace the current fixed robotic skill set.

Check for updates

Academic Editors: Victor García, David Zorrilla and Jesús Sánchez-Márquez

Received: 5 August 2026
Revised: 23 September 2026
Accepted: 25 September 2026
Published: 2 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Keywords: neuro-symbolic architecture; multimodal large language models; hierarchical task network; behavior tree; large language models

1. Introduction

Vision-Language-Action (VLA) models have moved robotics closer to general-purpose embodied control by coupling visual understanding, language conditioning, and action generation within a single policy [1]. Models such as RT-2 [2], OpenVLA [3], and $\pi_{0.7}$ [4] serve as a read/write API to physical reality, taking natural language commands and real-time visual observations to directly output executable low-level actions. This paradigm promises to reduce task-specific engineering, shifting the field from rigidly programmed robotic systems, historically dependent on intensive Offline Programming, toward fully cognitive and adaptive frameworks [5]. However, these models demonstrate excellent zero-shot transfer capabilities in unstructured household or general-purpose environments, their direct transplantation into actual industrial manufacturing and assembly exposes critical architectural limitations [6].

The difficulties derive from the conflict between the probabilistic neural architectures of VLA and the strict, deterministic requirements of manufacturing processes. Because VLAs map visual and linguistic inputs directly to action tokens based on statistical plausibility, they inherently lack the deductive semantic reasoning needed to natively process structured industrial artifacts like Computer-Aided Design (CAD) models, Bills of Materials (BOMs), and formal process plans [7]. Furthermore, standard VLAs relying on 2D visual encoders suffer from a spatial reasoning gap that hinders the high-precision, zero-tolerance 3D manipulation required for complex tasks like multi-pass welding or intricate assembly [8]. Finally, these models struggle significantly with long-horizon, dependency-heavy tasks, as their token-generation mechanisms cannot reliably maintain temporal consistency, manage prerequisite task completion, or execute robust error recovery without the integration of external hierarchical planning frameworks [9].

Based on these limitations, four Research Questions (RQs) are defined in this paper for the industrial deployment of VLA.

- RQ1. How can natural language instructions be robustly translated into formal, deterministic assembly planning directives through the integration of factory-specific data?
- RQ2. What mechanisms are required to maintain state tracking and execution precision across long-horizon industrial assembly, preventing the degradation of multi-step sequences?
- RQ3. How can external symbolic verification be utilized to ground generative outputs, thereby enforcing strict physical constraints and detecting and removing hallucinations?
- RQ4. To what extent can reactive execution architectures, coupled with real-time closed-loop vision systems, support bounded, predefined fault handling during industrial assembly, and what is required to extend this toward autonomous detection and recovery from unforeseen physical disturbances?

To address these RQs, the paper introduces a novel neuro-symbolic architecture tailored to the domain of industrial automation, with particular emphasis on small-batch robotic assembly. In this architecture, an ontology-grounded planning layer generates a Hierarchical Task Network (HTN) and an executable Behavior Tree (BT) from technical process data, namely CAD, process plan, BOM and technological constraints. The HTN provides long-horizon decomposition and allocation logic [10], and the BT provides modular, reactive, failure-tolerant execution semantics [11]. A fixed robotic skill library, informed by multi-modal LLM-based perception, is restricted to short-horizon embodied leaves

of the BT, such as local manipulation, alignment, insertion, and inspection segments. To make the approach reproducible, transferable, and auditable across plants, the planning layer is grounded in an industrial ontology and related semantic resources [12], with validation through competency questions, SPARQL checks, and explicit plan-to-execution correspondences [13].

The workflow transplants into an industrial pipeline different concepts proposed in recent literature about neuro-symbolic logical Artificial Intelligence (AI), semantic ontology-grounded generative AI and hybrid deliberative-reactive robot control. The pipeline is built on four layers: generation of a semantic-grounded planning problem respective of technical, logical and physical constraints, hierarchical planning of the assembly tasks, reactive execution inside Human-In-The-Loop (HITL) error recovery procedure [14], and short-horizon execution through a fixed robotic skill library informed by multi-modal LLM-based perception.

The paper presents this workflow as a feasibility demonstration and argues, based on system-level results and preliminary qualitative observations, that adherence to it is associated with a reduction in uncertainty, an improvement in process quality, and an effective mitigation of hallucination phenomena. At present, these outcomes can be achieved in a manner consistent with industrial standards only under the supervision of a human operator. Such human oversight eliminates the necessity for model fine-tuning. Nevertheless, the reliance on human intervention can be progressively diminished through the continuous updating of the interaction history via the bounded-memory adaptation model [15].

Section 2 reviews the state of the art in dual-system VLA architectures and highlights the unresolved challenges associated with their application to industrial processes. Section 3 presents the proposed neuro-symbolic framework. Section 4 reports the results obtained from its application to selected case studies, discussing the issues encountered during the initial implementation and the corresponding solutions adopted, including the incorporation of HITL mechanisms. Finally, Section 5 synthesizes the overall findings and outlines the remaining open challenges.

2. State of the Art and Challenges of Dual-System VLA

A VLA model is formally defined as a multi-modal transformer architecture that ingests visual observations alongside natural language instructions and directly outputs low-level robotic control commands. By treating robotic motor commands as a sequential language, tokenizing physical actions in a manner analogous to how Large Language Models (LLMs) tokenize text, VLAs establish deep, shared representations across visual perception, semantic understanding, and motor behavior [16]. To balance the conflicting demands of deep, slow semantic reasoning and high-frequency, rapid motor control, the field is increasingly adopting dual-system architectures inspired by human cognitive frameworks [17]. System 2 comprises a foundation model responsible for high-level perception of the environment and handling image observations alongside text instructions at a slower rate. System 1 is a lightweight, high-frequency control policy that generates reactive motor actions based on the guidance communicated from System 2.

The application of dual-system VLA architectures to industrial contexts can leverage the extensive body of knowledge in control systems developed for industrial tasks, thereby enabling a relatively smooth deployment of System 1. However, it poses significant challenges for System 2, which are typically neglected in other robotic applications. These challenges include the effective utilization of process data, the technological grounding of the action plan, the detection and mitigation of hallucinations, the management of long-horizon tasks, and the identification and correction of errors.

https://doi.org/10.3390/ai7100395

2.1. Inability to Leverage Structured Industrial Data

Current VLA architectures extract high-level scene embeddings and local geometric features from standard 2D and 3D visual inputs [18]. They use this visual information to guide basic manipulation based on high-level textual prompts, such as "put the red ball in the blue box". However, industrial operations could not be guided by visual intuition and casual commands. They are governed by rigorous, highly structured technical artifacts that must be executed exactly.

A fundamental limitation of the current VLA state-of-the-art is the inability to natively ingest and align industrial process data into the multi-modal prompting pipeline [19]. To bridge this gap, structured data must be systematically integrated into the model's inputs. For instance, CAD models should be integrated directly into the visual prompt to bridge the inherent spatial reasoning gap. Instead of relying on depth estimates derived from 2D pixels, CAD integration provides the VLA with exact topological volumes, spatial affordances, and zero-tolerance geometric grounding. Simultaneously, Engineering BOMs and formal process plans must be integrated into the textual prompt to enforce unyielding constraints regarding material properties, mandatory tooling, and sequencing logic. Currently, because VLAs map raw camera pixels and basic language directly to action tokens, they lack the symbolic reasoning capacity to ground their physical actions in strict tolerances or industrial ontologies [20].

2.2. Grounding Robotic Tasks in Industrial Procedures

The capability of an industrial VLA model must be evaluated through its application to tasks that require seamless, fault-tolerant synthesis of cognitive reasoning, and continuous, high-precision physical control. The complex industrial operations of welding, gluing, and handling present unique challenges that expose the boundaries of current artificial intelligence frameworks and necessitate the integration of formal process planning [21,22].

While basic pick-and-place operations are the standard benchmark for proving generalist VLA capabilities, complex industrial assembly requires an advanced understanding of physical affordances, temporal sequences, and force-torque feedback. The robot must not only recognize the target object but must also infer how that object is intended to fit into a larger sub-assembly, respecting constraints such as fragility, geometric orientation, and insertion force [23].

To translate an industrial process plan into a consistent operating plan for the robot we need to satisfy three levels of grounding. The first is the semantic grounding to assure a semantic valid sequence of high level tasks (e.g., 'Fetch Bracket_A', 'Align Bracket_A to Chassis', 'Fasten Bolt_1'). The second is the logical grounding to ensure that the state-space constraints are respected (e.g., robots cannot execute 'pick Bolt_1' when 'gripper_status' == full). Finally, we must ensure that the robot can physically execute the logical step in 3D space, by checking reachability, collision avoidance, joint limits, and payload capacities (Geometric & Kinematic Grounding).

2.3. The Threat of Industrial Hallucinations

Small-batch manufacturing environments—such as custom machining cells, specialized printed circuit board assembly lines, or aerospace composite layups—are characterized by high product variance, unstructured workspaces, and low production volumes [24]. These constraints fundamentally preclude the use of long, exhaustive parameter fine-tuning, massive data collection campaigns, or reinforcement learning regimens that typically align robotic foundation models to specific tasks. In these high-variance settings, a robotic system must frequently operate in a zero-shot or few-shot capacity. When deployed under such conditions, VLA models are highly susceptible to phenomena analogous to the halluci-

nations observed in purely textual LLMs. In this context, however, these errors do not merely result in incorrect text but manifest as catastrophic physical failures within the manufacturing cell.

The literature does not yet converge on a standard taxonomy, so we make a synthesis across three neighboring studies: visual grounding failures in VLMs, scene-task inconsistency failures in embodied planning, and physically invalid or infeasible actions in generative robot policies. A useful taxonomy for robotics is therefore as follows:

- Visual hallucination. The model asserts an object, attribute, relation, affordance, or spatial target that is not supported by the current sensory evidence. In the literature, this appears as an object, attribute or multi-modal hallucination and visually ungrounded generation [25]. In embodied settings, it also includes false localization or target selection. Multi-modal mutual-information decoding is proposed to reduce multi-modal hallucinations during inference by maximizing the mutual information between the text generated and the corresponding visual context [26].
- Textual hallucination. The model produces unsupported language-side content: a wrong decomposition of the user request, a symbolic goal that does not match the scene, an infeasible subtask, a synonym substitution that breaks reference consistency, or a counterfactual presupposition accepted as fact [27]. Embodied agents hallucinate under scene-task inconsistencies and often fail to reject infeasible tasks [28]. Many VLMs also fail on counterfactual presupposition questions even when object-perception mitigation exists [29].
- Physical hallucination. The policy produces or commits to actions that are physically invalid, unsafe, or causally ineffective despite appearing plausible in model space. Recent theory explicitly names this as action hallucination, arguing that generative VLA policies can fail because of topological, precision, and horizon barriers. In embodied planning terms, this also includes "doing something concrete but wrong" in response to an impossible task rather than abstaining [30].

Let x denote the natural-language instruction, O_t the multi-modal observation at time t, s_t a latent world state, and a_t the executed action. A generic VLA or VLA-centered controller induces a conditional rollout distribution:

$$p_\theta(\tau, z_{1:L} \mid x, O_{1:T}) = \prod_{t=1}^{T} p_\theta(z_t \mid h_t)\, p_\theta(a_t \mid z_{\le t}, h_t)\, p(s_{t+1} \mid s_t, a_t), \tag{1}$$

where $h_t = (x, O_{\le t}, a_{<t})$ and z_t denotes a latent variable such as plan steps or action tokens. This factorization covers direct VLAs, planner-plus-policy systems, and newer dual-model systems in which embodied reasoning and action generation are separate but coupled.

A useful hallucination definition is oracle-based: a hallucination occurs when model outputs disagree with a chosen external oracle of validity. In language-vision work the oracle is visual grounding; in embodied planning it is scene-task consistency; in robotics control it is feasibility, safety, or task completion. Using that oracle view, we define three hallucination probability indicators:

$$H_t^{vis} = \mathbf{1}\{\hat{y}_t \notin \mathcal{G}(O_t, s_t)\}, \tag{2}$$

$$H_t^{txt} = \mathbf{1}\{\neg \mathrm{Entail}(\hat{z}_t; x, \mathcal{G}(O_{\le t}, s_t), \mathcal{A}_{\mathrm{skill}})\}, \tag{3}$$

Note: the negation $\neg$ was added to correct the polarity of the indicator, so that $H_t^{txt} = 1$ when the generated content is not entailed by the grounded context—i.e., when a textual hallucination occurs—consistent with the convention used in Equations (2) and (4).

$$H_t^{phy} = \mathbf{1}\{a_t \notin \mathcal{A}_{\mathrm{safe}}(s_t) \vee c(s_t, a_t) > 0\}, \tag{4}$$

where $\mathcal{G}$ is a sensor-grounded set of valid scene assertions, $\mathcal{A}_{\text{skill}}$ is the admissible symbolic action or skill vocabulary for the agent, $\mathcal{A}_{\text{safe}}(s_t)$ is the state-dependent safe action set, and c is a constraint-violation function such as collision, force-limit exceedance, self-contact, forbidden-zone entry, or task-infeasibility certificate. This tripartite decomposition is a synthesis, but it matches the empirical structure seen in VLM grounding, embodied hallucination studies, and action-hallucination theory.

A probabilistic detector estimates

$$q_\phi(H = 1 \mid r_t), \tag{5}$$

where r_t is a vector of runtime risk signals. In deployment, q_ϕ can be a learned detector, a conformal score, a rule-based risk combiner, or a hybrid of the three. FAIL-Detect effectively operationalize this idea by transforming policy behavior into scalar risk signals and thresholding them with conformal calibration [31].

2.4. Poor Management of Long-Horizon Tasks

Despite the emergence of dual-mode VLA systems, they remain weak on multi-step, dependency-heavy execution. The literature explicitly frames most existing VLA frameworks as primarily short-horizon, System 1 models, highly capable of fast, reactive motor control, but severely limited in deliberative, long-horizon manipulation [32].

Industrial assembly requires an advanced understanding of physical affordances, temporal sequences, and prerequisite task completion. Without explicit planning abstractions, the underlying VLA relies purely on its training weights to guess the sequence of actions. For example, in a complex multi-part assembly like a motor housing, a VLA might successfully execute the gross motion of retrieving a component but fail to recognize that a necessary subtask—such as applying a thermal sealant—must occur before final insertion. This frequently leads to temporal misordering or severe logical errors. The combinatorial complexity of sequencing atomic skills over a long horizon overwhelms the finite memory and attention mechanisms of standard autoregressive VLAs, leading to compounding errors, spatial drift, and an eventual collapse of the execution pipeline.

The literature recognizes that long-horizon execution benefits from explicit planning abstractions. SayCan argued that language models require real-world grounding through available skills and affordances [33]. PROGPROMPT showed that program-like prompt structures, explicit action lists, and object inventories improve situated plan generation [34]. More recent work such as Nl2Hltl2Plan translates natural language into hierarchical intermediate structures before solving with off-the-shelf planners [35]. ChatHTN interleaves LLM-generated decompositions with symbolic HTN planning while remaining provably sound [36]. HBTP uses LLM reasoning only as a heuristic aid while keeping BT planning as the reliable backbone [37]. Across these works, the common lesson is that free-form language generation becomes more dependable when it is constrained by formal task structures, explicit action vocabularies, and symbolic validation.

2.5. Inadequate Error Correction and Reactivity

Standard VLA models exhibit poor error correction and limited real-time reactivity to local disturbances. Industrial environments are highly dynamic, requiring systems to constantly monitor guards, retry local actions, switch to fallbacks, or safely pause for human confirmation when anomalies occur [38].

Because current VLAs generate actions based on generalized probabilistic patterns rather than explicit state-machine logic, they lack the modular, reactive, and failure-tolerant execution semantics required for factory floor deployment. When a minor visual occlusion (such as a passing shadow) or a physical misalignment (such as a slipped gear) occurs, the

long-horizon execution is prone to cascading failures. A monolithic VLA policy lacks an explicit memory of completed subgoals and easily overfits to immediate visual features. Thus, simply re-invoking a failed skill policy without explicit structural recovery mechanisms often leads to the robot repeating the exact same flawed motion. Without the integration of deterministic hierarchical planning (such as HTN) or reactive execution layers (such as BTs), VLA models cannot reliably isolate a local error, dynamically replan around a global disturbance, or guarantee a safe and auditable recovery.

3. Materials and Methods

It is necessary to stress that the proposed neuro-symbolic solution is not a complete, fully autonomous architecture. Rather, it operates strictly as a mitigation strategy for the operational risks, spatial reasoning gaps, and logical hallucinations discussed in Section 2. Because current VLA and LLM architectures cannot guarantee absolute deterministic safety in unconstrained mechatronic environments, the planning layer within the proposed pipeline must pass through a HITL validation mechanism prior to physical deployment on the factory floor. The proposed framework deploys a multi-layered, tool-augmented pipeline designed to synthesize executable robotic control directly from unstructured assembly intent. The architecture is constituted by three functional layers: scenario analysis (knowledge layer), task planning (reasoning layer), and orchestration (execution layer), as shown in Figure 1.

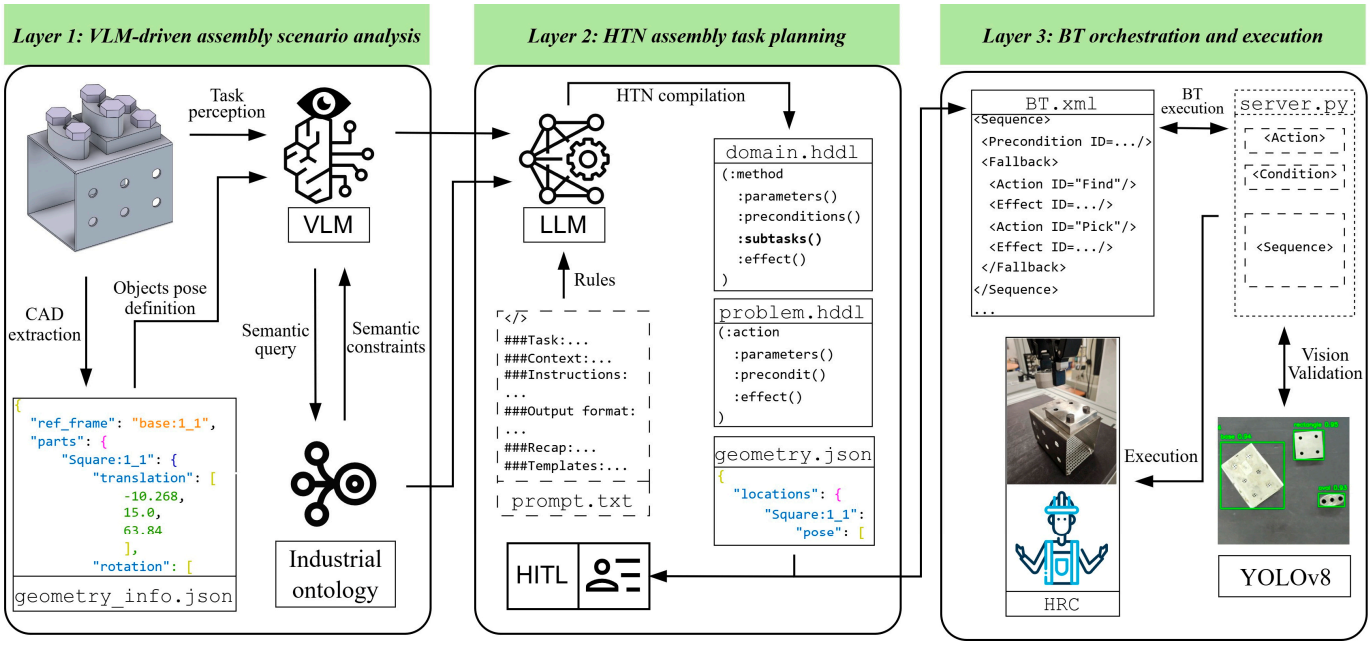


Figure 1. The neuro-symbolic framework for multimodal LLM deployment in industrial assembly.

3.1. Layer 1: VLM-Driven Assembly Scenario Analysis

The first layer establishes the multi-modal perception and deterministic knowledge foundation of the system. To prevent geometric hallucinations, generative reasoning is anchored by two specialized external modules prior to any planning sequence.

- CAD Extraction Module: This tool processes Standard for the Exchange of Product Model Data (STEP) files using the Open CASCADE Technology framework [39]. It extracts exact spatial features, computing a homogeneous transformation matrix $T \in SE(3)$ for each component. The module generates an axis-aligned bounding box defined by $dx = x_{max} - x_{min}$, $dy = y_{max} - y_{min}$, and $dz = z_{max} - z_{min}$. This deterministic spatial data is serialized into a JSON artifact.

- Ontological Semantic Modeling: The numerical geometric data is translated into a formal machine-interpretable knowledge representation using the Web Ontology Language (OWL) [40]. Physical components are classified as instances of iof-core:MaterialArtifact [41], populating an Assertional Box (A-Box) that is serialized in Turtle format.
- HITL Validation Gate 1: Before advancing to task generation, a human operator must review the extracted CAD coordinate frames and the generated ontological semantic triples to verify alignment with the physical workspace.

3.2. Layer 2: HTN Assembly Task Planning

The second layer constitutes the cognitive reasoning core, where a highly constrained LLM translates the validated state space into a formal HTN.

- Agentic Prompting: The LLM receives a programmatic prompt containing the ontological limits and the exact geometric arrays extracted in Layer 1. To enforce deterministic behavior and minimize factual drift, strict inference hyperparameters are applied (temperature $= 0.1$, $top_k = 5$, and $top_p = 0.2$).
- Hierarchical Domain Definition Language (HDDL) Generation: The agent synthesizes the assembly logic into the HDDL [42]. This formalizes the domain methods, predicates (e.g., (part-at ?p ?l)), and collaborative constraints necessary for the sequence.
- HITL Validation Gate 2: The generated HDDL schemas (domain and problem files) must be manually audited by a systems engineer to confirm that the proposed topological constraints and hierarchical decompositions are mechanically feasible.

3.3. Layer 3: BT Orchestration and Execution

The declarative HDDL plan is ultimately transformed into a continuous-time, reactive execution policy.

- BT Synthesis: A deterministic compilation script parses the HDDL syntax, mapping HTN methods to BT Sequence subtrees and primitive actions to executable Action nodes (e.g., Pick, Place). The output is a highly structured XML file.
- Reactive Control Loop: Execution is governed by a tick-based control model running at 20 Hz. The BT manages non-blocking TCP/IP socket communications with the physical robot controller (e.g., TMflow). Furthermore, an external validation server utilizes YOLOv8 to process real-time RGB video streams, confirming component presence and spatial bounding box overlaps before and after manipulation [43].
- HITL Validation Gate 3: During execution, a human operator remains engaged in the control loop, utilizing the BT's collaborative Wait nodes to safely synchronize human-driven manual fastening sequences with robotic positioning tasks.

3.4. Implementation Details

The cognitive reasoning layer was implemented using the open-weights Gemma 4 E4B and Gemma 4 26b models, served locally via LM Studio v0.4.25 (Q4_K_M quantization) and queried through the standard openai Python v3.19 client redirected to a local endpoint, with no external API calls. Decoding used temperature = 0.1, top_k = 5, top_p = 0.2. HDDL-to-BT compilation was performed by a custom, regex-based deterministic compiler applying two-tier validation (XML Schema Definition syntactic validation and Abstract Syntax Tree-based semantic validation); no external search-based HTN solver (e.g., SHOP2, PANDA) was deployed in the current implementation, and the "symbolic planner" referenced elsewhere in this paper denotes this deterministic rule-based validator/compiler rather than a search-based planner. Object detection used a YOLOv8n model, fine-tuned by the authors on a custom-labeled dataset of the three component classes used in the experimental tasks

(oval, rectangle, base), with a detection confidence threshold of 0.5, using a Logitech (Logitech International S.A., Lausanne, Switzerland) C505 HD camera. Robotic actions were issued as discrete commands (e.g., MOVE_TO_TARGET, FIND_<OBJECT>) over a custom TCP/IP protocol to the OMRON (OMRON Corp., Kyoto, Japan) TMflow controller, orchestrated within a py_trees-based Behavior Tree ticking at 20 Hz; the underlying motion primitives are implemented within the proprietary TMflow environment and are therefore not exposable as source code.

4. Results

To empirically evaluate the architecture's capacity to autonomously generate and execute collaborative sequences, a physical testing environment was established at the Mind4Lab facility. The key data and code used to conduct the experiments can be found in the Supplementary Materials.

4.1. Experimental Setup

The execution testbed featured an OMRON TM5-900 collaborative manipulator equipped with a Robotiq (Robotiq Inc., Lévis, QC, Canada) 2F-85 gripper, interacting with a human operator over a centralized workbench, as shown in Figure 2. Figure 3 shows the CAD representation and physical execution for all the tasks. The cognitive reasoning layer was evaluated across two distinct hardware configurations to assess scalability: a lightweight baseline using the Gemma 4 E4B model (on an NVIDIA (NVIDIA Corp., Santa Clara, CA, USA) RTX 4070) and a higher-capacity Gemma 4 26B model (on an NVIDIA RTX 5090). Local models are adopted to assure data protection.



Figure 2. The TM5-900 robot with Robotiq 2F-85 gripper.

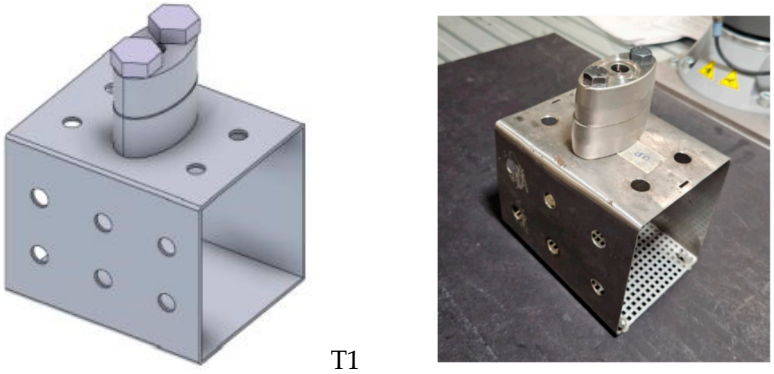


Figure 3. *Cont.*

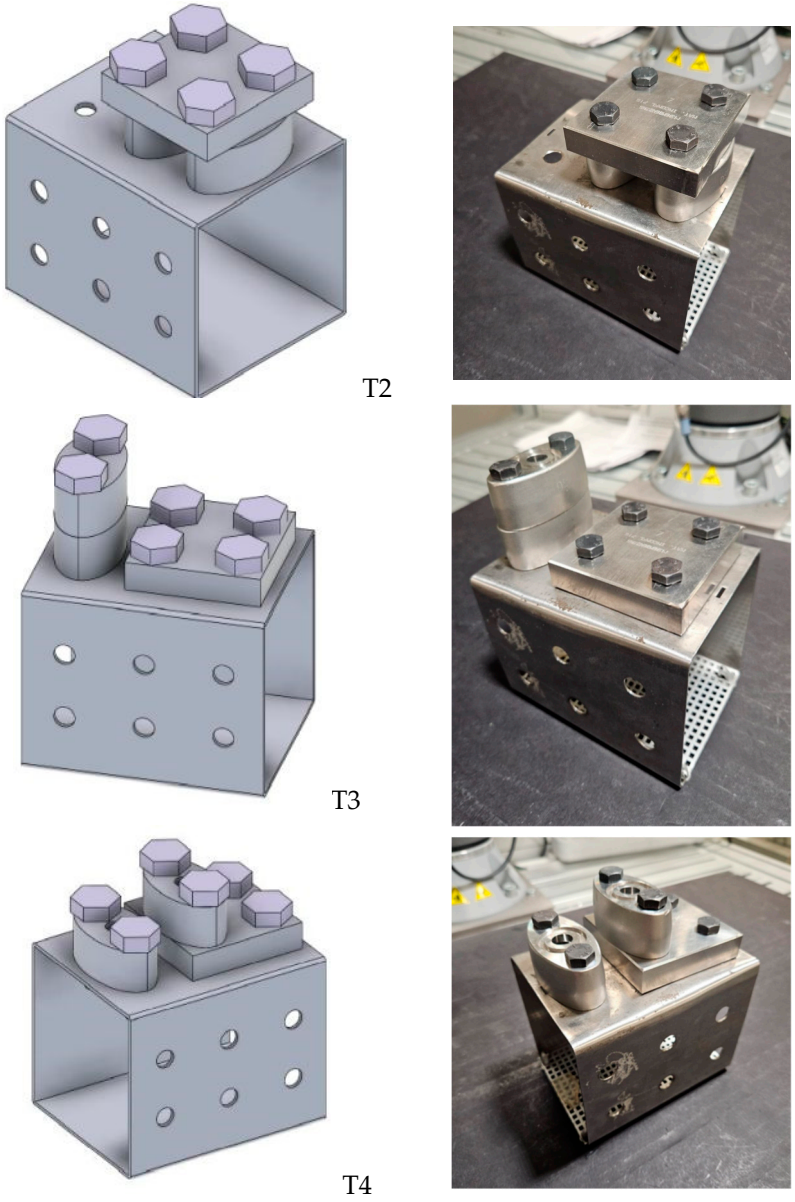


Figure 3. The CAD representation and physical execution for tasks T1, T2, T3, T4.

The system was tasked with 120 experimental executions per model configuration, distributed evenly across four assembly configurations of escalating topological complexity (Task 1 to Task 4), corresponding to 30 trials per task for each of the two Gemma configurations (240 trials in total). Both the Gemma 4 E4B and Gemma 4 26b configurations were evaluated on the physical robotic platform; the E4B configuration served as the primary diagnostic baseline for analyzing planning and execution failures, while the 26b configuration was evaluated as the higher-capacity comparison, which produced no execution-level failures beyond those already present at the planning stage.

The four tasks used a common assembly kit consisting of one base, one rectangular plate, and two oval brackets. T1 involved mounting the two ovals sequentially on the base. T2 required the two ovals to be mounted side-by-side before mounting the rectangular plate above them. T3 retains the stacked-oval configuration but introduces a front/back relationship between the oval assembly and the plate. T4 required the first oval to be mounted on the top-front region, followed by the rectangular plate on the top-back region and the second oval above the rectangle, aligned with the base center. All four tasks share the same underlying primitive action vocabulary; task complexity increases primarily through

increasingly constrained spatial and topological dependencies and deeper hierarchical decomposition, rather than through new primitive actions.

The human interventions required by the experimental protocol were planned collaborative actions —specifically, the manual fastening of components after the robot had positioned and held them—explicitly represented through the collaborative Wait nodes of the Behavior Tree and were therefore not counted as unplanned interventions in the metrics reported below. The human validation gates (Sections 3.1–3.3) did not constitute an experimental manipulation or an automatic replanning mechanism: Gate 1 verified the CAD/ontology grounding, Gate 2 consisted of a manual audit of the generated HDDL domain and problem files, and Gate 3 maintained human supervision during physical execution; invalid plans were not systematically rewritten by the operator after validation.

4.2. Experimental Metrics and Validation Protocol

To systematically evaluate the architecture and isolate the quality of the LLM-driven symbolic planning phase from the robustness of the physical execution pipeline, the validation protocol was divided into two primary domains: cognitive planning quality and system performance and execution robustness. Given that the solution is a mitigation strategy rather than a fully autonomous system, the metrics explicitly account for mandatory HITL cooperative actions while penalizing unplanned interventions.

4.2.1. Cognitive Planning Quality

This domain quantifies the capability of the tool-augmented pipeline to synthesize coherent, executable, and grounded HTNs.

- Plan Validity Rate (R_{valid}): This metric verifies whether the generated logic can be successfully parsed and executed by the symbolic planner. A generated plan is categorized as valid strictly if the HDDL syntax is correctly formatted, all non-primitive tasks are fully decomposable into executable primitives, and no undefined symbols or fictitious predicates are invoked. It is calculated as:

$$R_{valid} = \frac{N_{valid\ plans}}{N_{total\ generated\ plans}}, \tag{6}$$

- Hallucination Rate (R_{hal}): This metric indicates the frequency of constraint violations, serving as a direct measure of the anti-hallucination boundaries enforced by the deterministic tools. A hallucination is formally defined as the generation of non-existing mechanical objects, physically impossible kinematic actions, or incorrect geometric configurations. The metric is computed at the planning-trial (plan) level: a trial is counted as hallucinated when the generated plan contains at least one identified hallucination, using the 30 planning trials performed for each task as the denominator. It is calculated as:

$$R_{hal} = \frac{N_{invalid\ elements}}{N_{total\ generated\ elements}}, \tag{7}$$

- Planning Time (T_{plan}): This metric tracks the computational overhead required to formulate the plan, capturing the temporal latency from the initial task query to the final Behavior Tree compilation. This encompasses LLM inference, deterministic tool querying, and HDDL serialization.

4.2.2. System Performance and Execution Robustness

This domain transitions from the symbolic environment to physical operation, analyzing the behaviour of the complete execution pipeline during collaborative human–robot assembly.

- Failure Taxonomy: All observed failures were systematically categorized into seven classes to distinguish cognitive bottlenecks from execution-level faults in Table 1. Failures F1 through F4 represent planning errors (e.g., invalid syntax, wrong geometric reasoning), F5 represents perception failures, and F6 through F7 denote execution failures (e.g., BT deadlocks, robotic grasping faults).

Table 1. Failure taxonomy adopted during the experimental campaign.

Failure Code	Description	Pipeline Layer
F1	Invalid HDDL syntax generation	Planning Layer
F2	Incorrect task decomposition	Planning Layer
F3	Non-existing entities or predicates	Planning Layer
F4	Wrong geometric/topological reasoning	Planning Layer
F5	Object detection failure	Perception Layer
F6	BT execution deadlock or invalid transition	Execution Layer
F7	Robot grasping or motion execution failure	Execution Layer

- Task Success Rate ($R_{success}$): This acts as the ultimate metric of systemic operative viability for the end-to-end pipeline. A physical trial is registered as a success strictly if all mechanical parts are assembled to the correct target configuration, the robotic manipulator encounters no unrecoverable deadlocks, and absolutely no human intervention is required beyond the explicitly programmed HITL cooperative fastening phases. It is calculated as:

$$R_{success} = \frac{N_{successful\ executions}}{N_{total\ executions}}, \tag{8}$$

- Plan-Execution Consistency (C_{exec}): This metric validates the reliability of the HDDL-to-BT synthesis compiler, ensuring the transition from a static HTN to a reactive BT preserves logical integrity. A run is consistent if and only if every executed BT leaf node directly corresponds to a theoretically mapped HDDL action, no unexpected operative actions occur, and the physical execution order preserves the logical dependencies imposed by the initial HTN decomposition. This criterion was assessed qualitatively for each trial rather than aggregated into a separate quantitative metric.

4.3. Cognitive Planning Performance

- Baseline Configuration (Gemma 4 E4B): As shown in Table 2, for simple vertical stacking (Task 1), the model achieved a 93.33% R_{valid} with a deterministic planning time of 39.10 s. However, in Task 2, which introduced complex planar constraints, R_{valid} dropped to 66.67%, and the hallucination rate peaked at 30.00%.

Table 2. Cognitive Planning Performance-Gemma 4 E4B baseline.

Task ID	Mean $T_{planning}$ (s)	Std Dev (s)	R_{valid} (%)	R_{hal} (%)
T1	39.10	4.82	93.33	10.00
T2	46.01	21.86	66.67	30.00
T3	46.39	31.82	80.00	20.00
T4	50.76	18.56	83.33	16.67

- Scaled Configuration (Gemma 4 26b): As shown in Table 3, increasing the model's capacity substantially reduced spatial and topological hallucinations. Across the 120 trials performed with the 26b configuration, this configuration achieved near-zero task-

level hallucination rates, with 0.00%, 3.33%, 0.00%, and 6.67% for T1–T4, respectively, while maintaining high plan validity.

Table 3. Cognitive Planning Performance-Gemma 4 26b.

Task ID	Mean $T_{planning}$ (s)	Std Dev (s)	R_{valid} (%)	R_{hal} (%)
T1	38.99	1.57	100.00	0.00
T2	38.36	3.52	96.67	3.33
T3	31.94	0.75	100.00	0.00
T4	31.38	1.52	93.33	6.67

Because the two configurations were evaluated on different GPUs (RTX 4070 for the E4B baseline and RTX 5090 for the 26b configuration), the observed reduction in planning latency for the larger model cannot be attributed to model capacity alone and should be interpreted as hardware-specific.

4.4. System-Level Execution and Reliability

The execution metrics revealed a direct correlation between cognitive planning quality and physical execution success, as shown in Table 4 and Figure 4.

Table 4. System-Level Execution Metrics (E4B configuration).

Task ID	Success Rate (%)	Failures	Primary Failure Mode
T1	86.67	5	Non-Existent Entities
T2	66.67	12	Wrong Assembly Order/Topology
T3	76.67	7	Missing Object States
T4	83.33	5	Wrong Geometric Extraction

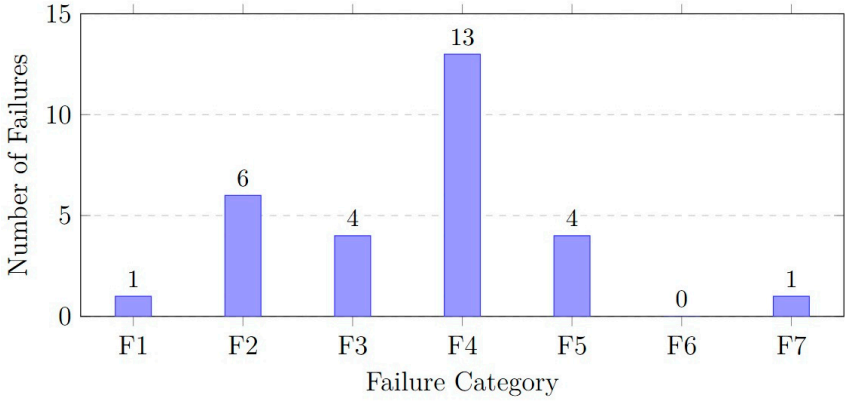


Figure 4. The distribution of observed failures across the experimental campaign (E4B configuration).

- Under the E4B configuration, the total execution success mirrored planning quality, with Task 2 achieving a 66.67% $R_{success}$.
- Analysis of the failure taxonomy revealed that geometric/topological reasoning errors (Category F4) constituted nearly half of all observed failures, while task decomposition failures (Category F2) also appeared with high frequency.
- Essentially, the reactive BT execution subsystem showed consistent stability across the evaluated tasks. No critical failures associated with BT transition deadlocks or invalid state propagation (Category F6) were recorded across the 120-trial campaign,

indicating that, once a validated plan passes the HITL oversight mechanisms, the reactive control layer performed reliably under the conditions tested.

Values in the "Failures" column denote failure occurrences (events) recorded per task, rather than the number of unsuccessful trials, since a single unsuccessful trial may involve more than one identifiable failure event; the reported success rates correspond to 4, 10, 7, and 5 unsuccessful trials (out of 30) for T1–T4, respectively, for a total of 29 failure occurrences across the campaign.

5. Discussion

The experimental validation confirms that integrating formal engineering knowledge into generative AI pipelines is a mandatory mitigation step, though not a comprehensive solution. By bounding the generative agent within deterministic parameters established by the CAD extraction and OWL ontologies, the architecture successfully prevented physical impossibilities that unconstrained VLAs typically hallucinate.

Prior to the main experimental campaign, informal preliminary trials were conducted to probe the necessity of symbolic grounding. Without any CAD/ontology grounding, the pipeline could not produce an executable plan at all, since no source of part dimensions or relative geometry was available to the model; this reflects a structural infeasibility rather than a hallucination in the strict sense. When CAD/ontology grounding was replaced with an assembled/exploded-view image and dimensioned technical drawings of the individual parts, the pipeline generated plans, but all trials failed physically. These observations motivated the architectural choice of formal CAD/ontology grounding but, given their limited scale and informal documentation, are reported here qualitatively rather than as a controlled ablation study; the present evaluation should accordingly be read as a system-level feasibility demonstration rather than a component-wise ablation.

5.1. Algorithmic and Generative Limitations

Despite the rigorous ontological grounding, the LLM remains the principal vulnerability within the architecture. The prevalence of F4 geometric failures under the baseline E4B model demonstrates that while the tools provide exact positioning coordinates, the LLM natively struggles with relative topological sequencing and inferring implied mechanical precedence. Because the ontology defines physical classification but lacks explicit kinematic constraints (e.g., alignment tolerances), the LLM is forced to deduce spatial relationships statistically. This confirms that the pipeline requires high-capacity models (such as the 26b variant) to securely process hierarchical dependencies. Furthermore, the HDDL parsing module relies on regex-based extraction strategies, which introduces severe fragility when handling complex nested constructs or partially ordered subtasks.

The nearly perfect performances of 26b model are due to the relatively simple tasks used in the experiment. More complex and realistic industrial assembly tasks would surely generate higher failure rates, requiring the scaling to more powerful models and an equivalent scaling of the hardware requirements. Nevertheless, because failures cannot be completely eliminated, HITL oversight remains necessary.

5.2. Perception and Execution Boundaries

The BT compilation strategy performed reliably across the evaluated tasks, supporting the choice to decouple static symbolic planning from continuous-time execution [44]. The asynchronous TCP/IP structure and memory-enabled sequence nodes allowed the robot to handle perception delays and human synchronization safely without stalling the control loop. However, the system's adaptability is bounded by a closed perception vocabulary. The YOLOv8 vision server and TMflow interface currently require manual pre-calibration

for each unique component class, preventing the system from dynamically accommodating zero-shot product variations.

5.3. Industrial Implications

The framework presents a viable pathway for High-Mix Low-Volume (HMLV) robotic assembly by drastically minimizing standard reprogramming overhead. Nevertheless, the results definitively underline that probabilistic AI reasoning, even when heavily augmented, is insufficient for isolated deployment in safety-critical manufacturing. Until generative systems can directly process native CAD alignment constraints and autonomously regenerate recovery HTN during global failure states, continuous HITL oversight remains an absolute industrial necessity.

6. Conclusions

The transition toward flexible HRC exposes the inherent limitations of traditional, rigid industrial programming. While VLMs and LLMs offer exceptional semantic adaptability for interpreting unstructured instructions, their probabilistic inference mechanisms introduce undesirable operational risks, frequently synthesizing physically impossible or structurally invalid actions.

This paper presented a multi-layered, neuro-symbolic framework designed to actively mitigate these risks by bounding generative AI within deterministic engineering constraints. By extracting exact geometric coordinates from CAD models and querying operational limits from formal OWL ontologies, the architecture is designed to anchor the LLM's reasoning process in physical reality, a design choice supported by both the main experimental campaign and the preliminary observations reported in Section 5. Furthermore, the deterministic compilation of HDDL schemas into reactive BTs in XML format effectively bridges the gap between abstract cognitive planning and robust robotic execution. Empirical validation utilizing an OMRON TM5-900 manipulator demonstrated that grounding quality directly dictates execution quality. When provided with verified semantic and spatial parameters, the BT execution subsystem remains highly stable and predictable.

However, this proposed architecture does not represent a complete, fully autonomous solution. It is strictly a mitigation strategy designed to minimize the hallucination phenomena and spatial reasoning gaps inherent in current generative models. Because probabilistic AI cannot yet guarantee absolute deterministic safety in unconstrained mechatronic environments, continuous HITL validation remains an absolute industrial necessity. Every layer of the pipeline—from semantic parsing to final robotic actuation—must pass through human oversight prior to actual deployment on the factory floor.

To transition this framework from an experimental prototype into a highly scalable, production-ready industrial solution, future research must address the current constraints across the cognitive, perceptual, and execution boundaries.

6.1. Cognitive and Semantic Enhancements

- Ontology Expansion: The semantic knowledge base must be broadened to include dynamic physical properties, such as material friction coefficients, force-torque thresholds, and center of mass shifts during assembly, allowing the planner to synthesize highly optimized manipulation strategies.
- Native CAD Integration: To eliminate spatial positioning errors entirely, future iterations should interface directly with native CAD assembly models rather than exported STEP files. This will allow the system to extract exact kinematic constraints and alignment rules programmatically.

- Domain-Specific Small Language Models (SLMs): Investigating the deployment of domain-specific SLMs fine-tuned on industrial CAD datasets will enable fully local deployment, improving data privacy and reducing computational latency [45].

6.2. Perception and Dynamic Adaptation

- Open-vocabulary Perception: The current reliance on YOLOv8 and pre-calibrated vision jobs restricts the system to a closed vocabulary. Integrating real-time, open-vocabulary VLMs directly into the high-frequency control loop would allow the system to dynamically identify novel components and evaluate assembly quality without manual pre-configuration.
- Active Dynamic Replanning: The architecture must evolve to support continuous feedback loops where the BT can autonomously feed the physical failure context back to the cognitive engine and query it to regenerate a partial HTN on the fly, enabling true dynamic, local replanning in response to environmental disturbances, thereby reducing reliance on manual human recovery.

6.3. Low-Level Execution and Control

- Real-time Middleware Integration: To bypass the latency introduced by TCP/IP string parsing and the constraints of proprietary interfaces like TMflow, the execution orchestrator should be directly integrated with real-time middleware such as ROS2 [46]. Utilizing protocols like EtherCAT would transition the framework to hard real-time control, enabling advanced compliance and collision avoidance.
- Learning-based Optimization: Integrating reinforcement learning at the BT leaf-node level would allow the manipulator to iteratively optimize its continuous control policies (e.g., minimizing trajectory jerk or refining grip force) while the HDDL layer continues to guarantee the macroscopic structural safety of the assembly sequence.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/ai7100395/s1, input data (task schemas, STEP files, BT schema), ontology (equipment), Python scripts used to ground the LLM, python planner agent.

Author Contributions: Conceptualization, D.A.; methodology, D.A.; software, L.U.; validation, D.A. and L.U.; formal analysis, D.A.; investigation, D.A. and X.Y.; resources, D.A.; data curation, D.A. and L.U.; writing—original draft preparation, D.A. and X.Y.; writing—review and editing, X.Y., X.L. and B.Y.; visualization, D.A. and X.Y.; supervision, D.A. and X.L.; project administration, D.A.; funding acquisition, X.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the China Scholarship Council Program (Grant No. 202506260086).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the Supplementary Material. Further inquiries can be directed to the corresponding author.

Acknowledgments: The authors acknowledge the use of the free version of Gemini-3.6-flash and DeepSeek-v4-flash for English language editing and proofreading. All intellectual content, figure design, scientific interpretation, and final manuscript content remain the sole responsibility of the author.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
A-Box	Assertional Box
BOMs	Bill of Materials
BT	Behavior Tree
CAD	Computer-Aided Design
HDDL	Hierarchical Domain Definition Language
HITL	Human-In-The-Loop
HTN	Hierarchical Task Network
IOF	Industrial Ontologies Foundry
LLMs	Large Language Models
OWL	Web Ontology Language
RQs	Research Questions
SLMs	Small Language Models
STEP	Standard for the Exchange of Product model data
VLA	Vision-Language-Action

References

1. Kawaharazuka, K.; Oh, J.; Yamada, J.; Posner, I.; Zhu, Y. Vision-language-action models for robotics: A review towards real-world applications. *IEEE Access* **2025**, *13*, 162467–162504. [CrossRef]
2. Brohan, A.; Brown, N.; Carbajal, J.; Chebotar, Y.; Chen, X.; Choromanski, K.; Ding, T.; Driess, D.; Dubey, A.; Finn, C.; et al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In Proceedings of the 7th Conference on Robot Learning, Atlanta, GA, USA, 6–9 November 2023.
3. Kim, M.J.; Pertsch, K.; Karamcheti, S.; Xiao, T.; Balakrishna, A.; Nair, S.; Rafailov, R.; Foster, E.P.; Sanketi, P.R.; Vuong, Q.; et al. OpenVLA: An Open-Source Vision-Language-Action Model. In Proceedings of the 8th Conference on Robot Learning, Munich, Germany, 6–9 November 2024.
4. Physical Intelligence; Ai, B.; Amin, A.; Aniceto, R.; Balakrishna, A.; Balke, G.; Black, K.; Bokinsky, G.; Cao, S.; Charbonnier, T.; et al. $\pi_{0.7}$: A Steerable Generalist Robotic Foundation Model with Emergent Capabilities. *arXiv* **2026**, arXiv:2604.15483.
5. Ma, Y.; Song, Z.; Zhuang, Y.; Hao, J.; King, I. A survey on vision-language-action models for embodied AI. *IEEE Trans. Neural Netw. Learn. Syst.* **2026**, *37*, 3031–3051. [CrossRef]
6. Kanta, S.; Tavassoli, M.; Chirkuri, V.T.; Kumar, V.A.; Punati, S.B.; Damacharla, P.; Katyara, S. Human centric general physical intelligence for agile manufacturing automation. *Adv. Eng. Inform.* **2026**, *72*, 104515. [CrossRef]
7. Li, W.; Zhang, R.; Shao, R.; Fang, Z.; Zhou, K.; Tian, Z.; Nie, L. SemanticVLA: Semantic-aligned sparsification and enhancement for efficient robotic manipulation. In Proceedings of the AAAI Conference on Artificial Intelligence, Singapore, 20–27 January 2026. [CrossRef]
8. Liu, Z.; Gu, Y.; Wang, Y.; Xue, X.; Fu, Y. ActiveVLA: Injecting Active Perception into Vision-Language-Action Models for Precise 3D Robotic Manipulation. *arXiv* **2026**, arXiv:2601.08325.
9. Jian, Y.; Tian, D.; Chen, X.; Wei, Z.; Liang, C.; Wang, M. PI-VLA: Adaptive symmetry-aware decision-making for long-horizon vision-language-action manipulation. *Symmetry* **2026**, *18*, 394. [CrossRef]
10. Georgievski, I.; Aiello, M. HTN planning: Overview, comparison, and beyond. *Artif. Intell.* **2015**, *222*, 124–156. [CrossRef]
11. Iovino, M.; Scukins, E.; Styrud, J.; Ögren, P.; Smith, C. A survey of behavior trees in robotics and AI. *Robot. Auton. Syst.* **2022**, *154*, 104096. [CrossRef]
12. Zhao, J.; Vogel-Heuser, B.; Ao, J.; Wu, Y.; Zhang, L.; Hartl, F.; Hujo, D.; Bing, Z.; Wu, F.; Knoll, A.; et al. Ontology Based AI Planning and Scheduling for Robotic Assembly. In Proceedings of the 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Abu Dhabi, United Arab Emirates, 14–18 October 2024. [CrossRef]
13. Bernardo, R.; Sousa, J.M.C.; Gonçalves, P.J.S. Ontological framework for high-level task replanning for autonomous robotic systems. *Rob. Auton. Syst.* **2025**, *184*, 104861. [CrossRef]
14. Li, C.; Zheng, P.; Li, S.; Pang, Y.; Lee, C.K.M. AR-assisted digital twin-enabled robot collaborative manufacturing system with human-in-the-loop. *Robot. Comput. Integr. Manuf.* **2022**, *76*, 102321. [CrossRef]
15. Nikolaidis, S.; Kuznetsov, A.; Hsu, D.; Srinivasa, S. Formalizing Human-Robot Mutual Adaptation: A Bounded Memory Model. In Proceedings of the 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), Christchurch, New Zealand, 7–10 March 2016. [CrossRef]

16. Zhong, Y.; Bai, F.; Cai, S.; Huang, X.; Chen, Z.; Zhang, X.; Wang, Y.; Guo, S.; Guan, T.; Lui, K.N.; et al. A Survey on Vision-Language-Action Models: An Action Tokenization Perspective. *arXiv* **2025**, arXiv:2507.01925.
17. Kortenkamp, D.; Simmons, R.; Brugali, D. Robotic Systems Architectures and Programming. In *Springer Handbook of Robotics*; Siciliano, B., Khatib, O., Eds.; Springer: Cham, Switzerland, 2016; pp. 283–306. [CrossRef]
18. Huang, J.; Zheng, H.; Ren, Y.; Wan, W.; Xu, X.; Polzer, J. Video2knowledge: Extracting temporally consistent task knowledge from monocular video for robot skill learning. *Robot. Comput. Integr. Manuf.* **2027**, *103*, 103371. [CrossRef]
19. Islam, B.U.; Prioli, J.P.J.; Azucena, J.C.H. Beyond rigid automation: A review of vision-language-action models for adaptive human-robot disassembly. *Robot. Comput. Integr. Manuf.* **2027**, *103*, 103360. [CrossRef]
20. Zappa, I.; Vignali, S.; Zanchettin, A.M.; Rocco, P. Symbolic representation of objects relative poses for robotic ma-nipulation tasks. *Eng. Appl. Artif. Intell.* **2026**, *163*, 112882. [CrossRef]
21. Lim, J.; Zhao, J.; Hernandez, E.; Kovalenko, I. Adaptive task planning and coordination in multi-agent manufacturing systems using large language models. *Robot. Comput. Integr. Manuf.* **2026**, *100*, 103245. [CrossRef]
22. Nguyen, E.-R.; Zhang, Y.; Ranasinghe, K.; Li, X.; Ryoo, M.S. Pixel Motion Diffusion Is What We Need for Robot Control. *arXiv* **2025**, arXiv:2509.22652.
23. Leanza, A.; Moroncelli, A.; Vizzari, G.; Braghin, F.; Roveda, L.; Spahiu, B. ConceptBot: Enhancing Robot's Autonomy through Task Decomposition with Large Language Models and Knowledge Graph. *arXiv* **2025**, arXiv:2509.00570.
24. Li, W.; Liu, X.; Du, X.; Shao, H.; Zhu, T.; Yang, S.; Yang, X. Matrix-structured manufacturing systems: Literature review and future directions. *Int. J. Prod. Res.* **2026**, 1–36. [CrossRef]
25. Bang, Y.; Cahyawijaya, S.; Lee, N.; Dai, W.; Su, D.; Wilie, B.; Lovenia, H.; Ji, Z.; Yu, T.; Chung, W.; et al. A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity. In Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), Nusa Dua, Bali, Indonesia, 1–4 November 2023. [CrossRef]
26. Favero, A.; Zancato, L.; Trager, M.; Choudhary, S.; Perera, P.; Achille, A.; Swaminathan, A.; Soatto, S. Multi-modal hallucination control by visual information grounding. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 16–22 June 2024. [CrossRef]
27. Zheng, Z.; Han, J.; Chen, K.; Cao, X.; Lu, X.; Lin, J. Translating regulatory clauses into executable codes for building design checking via large language model driven function matching and composing. *Eng. Appl. Artif. Intell.* **2026**, *163*, 112823. [CrossRef]
28. Chakraborty, T.; Ghosh, U.; Zhang, X.; Niloy, F.F.; Dong, Y.; Li, J.; Roy-Chowdhury, A.; Song, C. HEAL: An Empirical Study on Hallucinations in Embodied Agents Driven by Large Language Models. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, 4–9 November 2025. [CrossRef]
29. Agia, C.; Sinha, R.; Yang, J.; Cao, Z.; Antonova, R.; Pavone, M.; Bohg, J. Unpacking Failure Modes of Generative Policies: Runtime Monitoring of Consistency and Progress. In Proceedings of the 8th Conference on Robot Learning, Munich, Germany, 6–9 November 2024.
30. Soh, H.; Lim, E. Action Hallucination in Generative Vision-Language-Action Models. *arXiv* **2026**, arXiv:2602.06339.
31. Xu, C.; Nguyen, T.K.; Dixon, E.; Rodriguez, C.; Miller, P.; Lee, R.; Shah, P.; Ambrus, R.; Nishimura, H.; Itkina, M. Can We Detect Failures without Failure Data? Uncertainty-Aware Runtime Failure Detection for Imitation Learning Policies. *arXiv* **2025**, arXiv:2503.08558.
32. Song, X.; Chen, W.; Liu, Y.; Chen, W.; Li, G.; Lin, L. Towards Long-Horizon Vision-Language Navigation: Platform, Benchmark and Method. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 10–17 June 2025. [CrossRef]
33. 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* **2022**, arXiv:2204.01691.
34. Singh, I.; Blukis, V.; Mousavian, A.; Goyal, A.; Xu, D.; Tremblay, J.; Fox, D.; Thomason, J.; Garg, A. ProgPrompt: Program generation for situated robot task planning using large language models. *Auton. Robot.* **2023**, *47*, 999–1012. [CrossRef]
35. Xu, S.; Luo, X.; Huang, Y.; Leng, L.; Liu, R.; Liu, C. Nl2hltl2plan: Scaling up natural language understanding for multi-robots through hierarchical temporal logic task specifications. *IEEE Robot. Autom. Lett.* **2025**, *10*, 10482–10489. [CrossRef]
36. Avila, H.M.; Aha, D.W.; Rizzo, P. ChatHTN: Interleaving Approximate (LLM) and Symbolic HTN Planning. *arXiv* **2025**, arXiv:2505.11814.
37. Cai, Y.; Chen, X.; Mao, Y.; Li, M.; Yang, S.; Yang, W.; Wang, J. HBTP: Heuristic Behavior Tree Planning with Large Language Model Reasoning. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Atlanta, GA, USA, 19–23 May 2025.
38. Raeiszadeh, M.; Ebrahimzadeh, A.; Glitho, R.H.; Eker, J.; Mini, R.A.F. Real-time adaptive anomaly detection in in-dustrial IoT environments. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 6839–6856. [CrossRef]
39. Liu, H.; Wu, Z.; Yuan, S.; Wang, Y.; Dong, L. Design and implementation of a three-dimensional CAD graphics support platform for pumps based on open CASCADE. *Processes* **2023**, *11*, 2315. [CrossRef]

40. Zhang, L.; Lobov, A. Semantic web rule language-based approach for implementing knowledge-based engineering systems. *Adv. Eng. Inform.* **2024**, *62*, 102587. [CrossRef]
41. The Industrial Ontologies Foundry Core Ontology. Available online: https://github.com/iofoundry/Core#the-industrial-ontologies-foundry-core-ontology (accessed on 29 May 2026).
42. Höller, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; Alford, R. HDDL: An extension to PDDL for expressing hierarchical planning problems. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 9883–9891. [CrossRef]
43. Afifah, V.; Erniwati, S. YOLOv8 for object detection: A comprehensive review of advances, techniques, and applications. *Int. J. Adv. Comput. Inform.* **2025**, *2*, 53–61. [CrossRef]
44. Liu, Z.; Wu, S.; Luo, H.; Qin, D.; Fang, Y. BT-TL-DMPs: A Novel Robot TAMP Framework Combining Behavior Tree, Temporal Logic, and Dynamical Movement Primitives. *IEEE/ASME Trans. Mechatron.* **2026**, 1–12. [CrossRef]
45. Chun, K.; Lim, S.; Yeo, C.; Park, S.; Mun, D. Toward fully automated CAD-CAE integration through design feature recognition and small language models. *J. Comput. Des. Eng.* **2026**, *13*, 275–302. [CrossRef]
46. ROS 2. Available online: https://github.com/ros2 (accessed on 20 June 2026).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.