Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Build on Priors: Vision-Language-Guided Neuro-Symbolic Imitation Learning for Data-Efficient Real-World Robot Manipulation

Pierrick Lorang

pierrick.lorang@tufts.edu

Austrian Institute of Technology GmbH (AIT)

Johannes Huemer

Austrian Institute of Technology GmbH (AIT)

Timothy Duggan

Tufts University

Kai Goebel

Austrian Institute of Technology GmbH (AIT)

Patrik Zips

Austrian Institute of Technology GmbH (AIT)

Matthias Scheutz

Tufts University

Research Article

Keywords: Neuro-symbolic, Imitation Learning, Task and Motion Planning, Symbolic Planning, Skill Learning, Human-Robot Interaction, Real-World Robotics

Posted Date: April 7th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9254688/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Build on Priors: Vision–Language-Guided Neuro-Symbolic Imitation Learning for Data-Efficient Real-World Robot Manipulation

Pierrick Lorang[1,2*], Johannes Huemer[2], Timothy Duggan[1], Kai Goebel[2], Patrik Zips[2], Matthias Scheutz[1]

[1*]Department of Computer Science, Tufts University, Medford, MA, USA.
[2]Complex Dynamical Systems, Austrian Institute of Technology GmbH (AIT), Vienna, Austria.

*Corresponding author(s). E-mail(s): pierrick.lorang@tufts.edu;
Contributing authors: johannes.huemer@ait.ac.at;
timothy.duggan@tufts.edu; kai.goebel@ait.ac.at; patrik.zips@ait.ac.at;
matthias.scheutz@tufts.edu;

Abstract

Enabling robots to learn long-horizon manipulation tasks from a handful of demonstrations remains a central challenge in robotics. Existing neuro-symbolic approaches often rely on hand-crafted symbolic abstractions, semantically labeled trajectories or large demonstration datasets, limiting their scalability and real-world applicability. We present a scalable neuro-symbolic framework that autonomously constructs symbolic planning domains and data-efficient control policies from as few as one to thirty unannotated skill demonstrations, without requiring manual domain engineering. Our method segments demonstrations into skills and employs a Vision-Language Model (VLM) to classify skills and identify equivalent high-level states, enabling automatic construction of a state-transition graph. This graph is processed by an Answer Set Programming solver to synthesize a PDDL planning domain, which an oracle function exploits to isolate the minimal, task-relevant and target relative observation and action spaces for each skill policy. Policies are learned at the control reference level rather than at the raw actuator signal level, yielding a smoother and less noisy learning target. Known controllers can be leveraged for real-world data augmentation by projecting a single demonstration onto other objects in the scene, simultaneously enriching the graph construction process and the dataset for imitation learning.

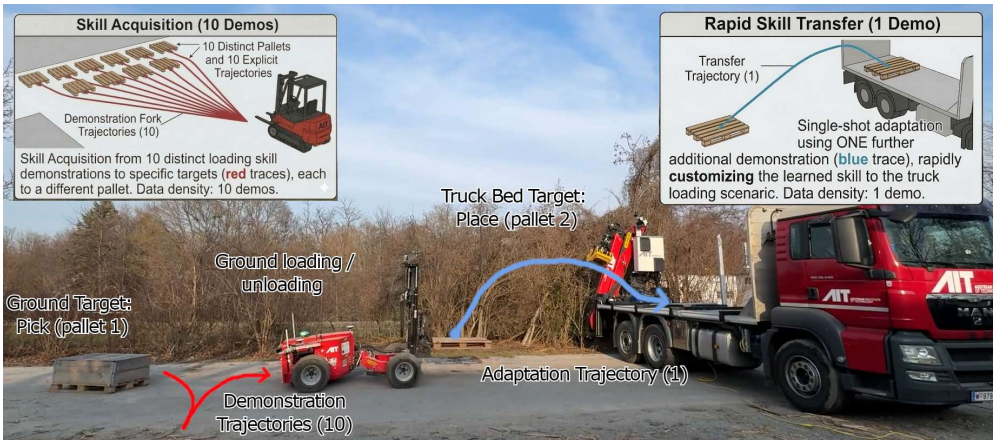


Fig. 1: Our forklift system loading two pallets from the ground and unloading them onto a truck. Such system builds on priors (perception, VLM for annotation and controls) to imitate long-horizon and complex task planning and acting from only 10 base skills demonstrations and 1 extra adaptation skill trajectory.

We validate our framework primarily on a real industrial forklift across statistically rigorous manipulation trials, and demonstrate cross-platform generality on a Kinova Gen3 robotic arm across two standard benchmarks. Our results show that grounding control learning, VLM-driven abstraction, and automated planning synthesis into a unified pipeline constitutes a practical path toward scalable, data-efficient, expert-free and interpretable neuro-symbolic robotics.

Keywords: Neuro-symbolic, Imitation Learning, Task and Motion Planning, Symbolic Planning, Skill Learning, Human-Robot Interaction, Real-World Robotics

1 Introduction

Teaching robots to perform complex, long-horizon tasks remains a fundamental challenge at the intersection of robotics, planning, and machine learning. Imitation learning has emerged as a compelling paradigm for acquiring robot behaviors from demonstrations, sidestepping the need for manual reward engineering (Hussein et al. 2017; Osa et al. 2018; Zare et al. 2024). Yet most imitation learning methods operate at the level of individual short-horizon skills, struggle with compounding distribution shifts over long execution horizons (Rajaraman et al. 2020; Xu et al. 2020), and require large demonstration datasets to generalize reliably. These limitations become particularly acute in real-world deployments, where tasks span many sequential steps, the number of available demonstrations is small, and the system must generalize to object configurations or task structures not seen during training. Humans address long-horizon problems naturally by abstracting continuous experience into symbolic structures—predicates, operators, and plans—that support flexible reasoning and

reuse (Newell and Simon 1976). Hierarchical approaches such as Task and Motion Planning (TAMP) (Wolfe et al. 2010; Kaelbling and Lozano-Pérez 2011; Garrett et al. 2021) mirror this strategy by decomposing problems into a symbolic planning layer and a continuous motion layer. However, they traditionally require hand-engineered symbolic domains, making them brittle and expensive to deploy in new settings. A natural direction is to learn symbolic models and low-level controllers directly from data. Prior work has made progress on learning symbolic abstractions (Konidaris et al. 2018; Bonet and Geffner 2020; Rodriguez et al. 2021; Ahmetoglu et al. 2022; Silver et al. 2022; Shah et al. 2024; Umili et al. 2021) or low-level policies (Illanes et al. 2020; Kokel et al. 2021; Guan et al. 2022; Silver et al. 2023; Goel et al. 2022; Lorang et al. 2024b,a, 2025b) independently, and some recent methods attempt to learn both jointly (Silver et al. 2023). This requirement not only demands significant domain expertise from engineers who must anticipate and encode the relevant symbolic vocabulary before deployment, but also renders the resulting systems brittle to novel environments where such prior knowledge is unavailable or incomplete. A further challenge that has received limited attention is the *data efficiency* of the full pipeline. Learning reliable controllers and meaningful symbolic abstractions simultaneously typically demands large demonstration datasets (Konidaris et al. 2018; Chitnis et al. 2022). This becomes particularly acute in settings that require behavioral flexibility, such as autonomous forklifts operating across varied warehouse layouts or collaborative manipulators adapting to new object configurations, where the closed-world assumptions that make industrial data collection tractable begin to break down, and systems must generalize from only a handful of examples. We address these challenges by proposing a unified neuro-symbolic framework that learns both a symbolic planning domain and low-level control policies from as few as one to thirty skill demonstrations per skill, without assuming any predefined symbolic vocabulary. Our key insight is that a Vision-Language Model (VLM) can replace the need for human annotations: it classifies demonstrated skills and identifies equivalent high-level states by comparing visual scene snapshots, enabling the automatic construction of a state-transition graph over the domain. This graph is passed to an Answer Set Programming (ASP) solver (Bonet and Geffner 2020; Rodriguez et al. 2021) that synthesizes a consistent PDDL planning domain (McDermott et al. 1998), capturing not only static spatial predicates, as clustering-based methods typically do (Shah et al. 2024), but also richer relational and temporal transitions such as waiting for a process to complete. The extracted PDDL domain drives an oracle function that automatically identifies, for each planning operator, the minimal relevant observation and action space, inferring the relevant object observation from operator groundings in the training demonstrations as an attention mechanism, and pruning the action space by filtering out dimensions that remain unused across those same demonstrations. The oracle additionally computes relative end-effector pose with respect to the object of interest, providing a compact geometric signal that further reduces learning complexity. Policies are trained at the *control reference level* rather than at the raw actuator signal level, yielding a smoother and less noisy imitation target. Controls notably enable real-world *data augmentation*: given a single demonstration, a skill can be projected onto alternative objects present in the scene, generating additional synthetic demonstrations that simultaneously enrich the

graph used for symbolic abstraction and expand the dataset for diffusion-based imitation learning (Mandlekar et al. 2023). We validate our framework on a real industrial forklift system and further demonstrate its generality by deploying the same pipeline on a Kinova Gen3 robotic arm across standard manipulation benchmarks. In both settings, the full pipeline, from raw demonstrations to interpretable long-horizon task execution, requires minimal data, no symbolic supervision, and generalizes robustly to novel configurations. This paper makes the following contributions:

- A VLM-driven graph construction pipeline that automates annotation, classifies skills and identifies equivalent high-level states from visual scene snapshots, requiring no predefined predicates, symbols, or manual engineering.
- An ASP-based symbolic abstraction method that synthesizes a PDDL planning domain from the state-transition graph, supporting rich relational and temporal operator structures.
- An oracle function that exploits the learned symbolic domain to isolate the minimal observation and action space per skill, including geometry-aware relative pose computation for unary operators.
- Control-level imitation learning with real-world data augmentation, enabling a single demonstration to be projected onto multiple scene objects to multiply effective dataset size for both graph construction and policy training.
- Experimental validation on a Kinova Gen3 arm and a real industrial forklift, demonstrating data-efficient, generalizable, and interpretable long-horizon task execution from as few as one demonstration per skill.

2 Related Work

2.1 Symbolic Planning and Neuro-Symbolic TAMP

Classical approaches to Task and Motion Planning (TAMP) decompose long-horizon robot behavior into a symbolic task planner operating over discrete abstractions and a continuous motion planner executing the resulting plan (Garrett et al. 2020, 2021). A large body of work has studied how to acquire the symbolic components of such architectures from data rather than manual specification.

One stream of research focuses on learning action models from symbolic traces, either from existing datasets or through environmental interaction (Chitnis et al. 2022; Kumar et al. 2023; Chitnis et al. 2022; Konidaris et al. 2018; Umili et al. 2021). Foundational work by Konidaris et al. (2018) established a principled link between low-level skills and abstract symbolic representations, showing that precondition and effect symbols are both necessary and sufficient for high-level planning. Building on this, Chitnis et al. (2022) introduced Neuro-Symbolic Relational Transition Models (NSRTs), which are data-efficient to learn and generalize over objects, and later extended this line to jointly learn symbolic predicates alongside operators (Kumar et al. 2023), removing the need for hand-specified state abstractions. A common assumption across all of these works is that symbolic predicates are either given or derived from supervised symbolic traces, as they do not emerge from raw, unlabeled continuous demonstrations.

An orthogonal direction reduces the data requirements for symbolic domain acquisition by framing the problem as combinatorial search. Bonet and Geffner (2020) showed that first-order symbolic representations for planning can be inferred from the structure of the state space via a SAT-based search, working from unlabeled state graphs rather than expert-annotated trajectories. This was later extended to Answer Set Programming (ASP) encodings (Rodriguez et al. 2021), enabling more expressive domain learning with minimal data. Prior work has also applied these solvers to learn continuous features that improve TAMP efficiency from a few example plans with embedded continuous data (Curtis et al. 2022); however, these approaches assume a predefined symbolic domain and solved plans as input.

In contrast to methods requiring predefined predicate vocabulary or expert-annotated symbolic traces, our framework requires only a human-provided skill name vocabulary and brief domain hints, automatically constructing the full predicate and operator structure via VLM-driven annotation and ASP synthesis. Unlike prior ASP-based approaches, we co-learn continuous controllers alongside the symbolic domain from a small number of demonstrations.

2.2 Reinforcement and Imitation Learning for Long-Horizon Tasks

A parallel line of research addresses long-horizon task execution through learning-based methods without explicit symbolic representations. Imitation learning approaches directly clone behavior from demonstrations: early work on skill sequencing (Manschitz et al. 2014; Le et al. 2018; Pertsch et al. 2021; Tanwani et al. 2021; Zhu et al. 2022; Teng et al. 2023) enables the imitation of multi-step behaviors but yields opaque policies that are difficult to repurpose for novel task structures. More recently, generative policy architectures have substantially improved the fidelity and dexterity of learned behaviors: Diffusion Policy (Chi et al. 2023) frames visuomotor control as a denoising diffusion process and captures complex multimodal action distributions, while Action Chunking with Transformers (ACT) (Zhao et al. 2023) predicts short action sequences to reduce compounding errors, achieving high success rates on fine manipulation from as few as fifty demonstrations. These methods were designed for single-skill or short-horizon settings and achieve impressive performance within that scope. Scaling such approaches to long-horizon compositional tasks, however, remains an open challenge that they were not designed to address: reusing sub-behaviors compositionally and adapting to novel task structures without retraining are complementary problems that our framework targets.

Reinforcement learning approaches have similarly been guided by pre-established symbolic domains to structure exploration (Icarte et al. 2020; Sarathy et al. 2021; Goel et al. 2022; Yang et al. 2018; Gehring et al. 2022; Cheng and Xu 2023). Reward machines (Icarte et al. 2020) decompose the reward function according to task structure, and symbolic oracles have been used to provide guidance or filter agent actions (Illanes et al. 2020; Mitchener et al. 2022). More recent hybrid frameworks (Sarathy et al. 2021; Goel et al. 2022; Lorang et al. 2024a, 2025b) spawn RL agents to bridge planning gaps when novel situations arise. However, these approaches

still rely on manually crafted symbolic representations and spend significant environment interaction time exploring to discover missing operators. Moreover, they evaluate primarily in discrete action domains, or in continuous settings where continuous controllers are assumed to exist rather than co-learned.

Our approach differs fundamentally: we co-learn both the symbolic domain and the continuous controllers from a small set of raw demonstrations, with no manual symbolic engineering and no RL exploration.

2.3 Learning Symbolic Abstractions from Raw Data

Extracting symbolic abstractions directly from raw trajectories is a more demanding challenge that has received comparatively less attention. Shah et al. (2024) proposed extracting relational symbols from raw demonstrations, but require fifty or more demonstrations to produce a functional planning domain, and the resulting operators are still refined using manually coded controllers. Segmentation methods (Loula et al. 2020) can ease this process by identifying sub-task boundaries, but do not resolve the underlying reliance on extensive data. Keller et al. (2025) cluster continuous observations to induce predicates, also requiring substantial data before stable abstractions emerge.

The approach most closely related to ours is that of Lorang et al. (2025a), which co-learns symbolic domains and low-level controllers from very few demonstrations. Our work builds on this paradigm and extends it with real-world data augmentation as well as automated annotation, demonstrating that the same pipeline, from raw demonstrations to interpretable long-horizon task execution, can be successfully deployed on a real forklift and generalized across manipulation benchmarks while requiring way less human input.

2.4 Relation to Vision-Language-Action Models

Vision-Language-Action models (VLAs) (Brohan et al. 2023; Black et al. 2024; Kim et al. 2024) represent a compelling alternative paradigm: by fine-tuning large pretrained vision-language models on robot trajectory data, they acquire policies that can be conditioned on natural language instructions and generalize across diverse tasks and object categories within their training distribution. Their expressiveness and language grounding make them attractive for real-world deployment. However, several fundamental limitations make end-to-end VLAs poorly suited to the setting we address.

Data requirements. VLAs are pretrained on datasets comprising hundreds of thousands to millions of demonstration trajectories (Brohan et al. 2023; Open X-Embodiment Collaboration 2023), and fine-tuning for new tasks typically requires hundreds of task-specific demonstrations (Kim et al. 2024). While broad pretraining can reduce fine-tuning requirements in well-represented domains, this advantage diminishes for specialized industrial settings such as autonomous forklift operation, which are absent from current VLA pretraining datasets and where our framework's data efficiency is most relevant. Our framework, by contrast, is designed explicitly

for the few-demonstration regime (1–30 skill demonstrations), making it applicable in industrial and specialized settings where large-scale data collection is impractical.

Lack of interpretability and replanning. VLAs are black-box function approximators: they produce actions directly from observations without exposing an intermediate symbolic representation of task state. Hence, there is no interpretable plan that an operator can inspect, no natural point at which to inject task-level constraints, and no mechanism for replanning if execution deviates from the intended task structure. Our framework produces an explicit PDDL plan before execution begins, which is human-interpretable and can be verified or overridden prior to deployment.

Compositional generalization. VLAs generalize by interpolation within their training distribution: they can adapt to new object instances or scene appearances, but composing skills in sequences not represented in training data remains an open challenge they were not designed to address (Black et al. 2024). Symbolic planning, by contrast, supports combinatorial generalization by construction: once a set of operators has been learned, any task expressible as a sequence of those operators can be planned and executed, even if that specific sequence was never demonstrated, contingent on the correctness of the learned PDDL domain, which is the primary technical challenge our framework addresses.

Complementarity. VLMs and VLAs are distinct: VLMs provide semantic understanding and annotation, while VLAs are end-to-end controllers mapping observations to actions. Our framework uses VLMs internally for skill classification, state equivalence judgment, and open-vocabulary detection, and the architectural comparison here is therefore between our neuro-symbolic pipeline as a whole and VLA-based end-to-end control, not between VLMs and VLAs. These approaches are complementary: the diffusion skill policies could be replaced by VLA-style controllers for tasks requiring richer visual conditioning, while retaining the symbolic planning layer for task-level generalization and interpretability. We leave this integration as a direction for future work.

3 Preliminaries: Neuro-Symbolic Planning-Diffusion Models

3.1 Symbolic Planning

Symbolic planning builds upon a formal domain description $\sigma = \langle \mathcal{E}, \mathcal{F}, \mathcal{S}, \mathcal{O} \rangle$, where $\mathcal{E}$ is a set of entities, $\mathcal{F}$ a set of boolean or numerical predicates over entities, $\mathcal{S}$ a set of symbolic states formed by grounded predicates, and $\mathcal{O}$ a set of operators. Each operator $o \in \mathcal{O}$ is defined by preconditions ψ and effects ω over predicates. A grounded operator $\hat{o}$ binds objects to parameters and can be applied if its preconditions hold, updating the state according to its effects. A planning task $T = (\mathcal{E}, \mathcal{F}, \mathcal{O}, s_0, s_g)$ seeks a plan $\mathcal{P} = [o_1, \ldots, o_{|\mathcal{P}|}]$ that transitions from initial state s_0 to goal state s_g (McDermott et al. 1998).

3.2 Diffusion Imitation Learning (IL)

Imitation learning (IL) aims to learn a policy $\pi(\tilde{s})$ from expert demonstrations $\{(\tilde{s}_t, a_t, \tilde{s}_{t+1})\}_{t=0}^{T}$, where $\tilde{s}_t$ denotes a continuous state, a_t the expert action, and $\tilde{s}_{t+1}$ the resulting state. The policy is trained by minimizing the mean squared error between predicted and expert actions: $\mathcal{L}(\pi) = \frac{1}{T}\sum_{t=0}^{T} \|\pi(\tilde{s}_t) - a_t\|^2$. Unlike reinforcement learning, IL bypasses exploration and reward engineering, providing a more data-efficient way to acquire complex behaviors directly from demonstrations.

Diffusion policies (Chi et al. 2023) are imitation learning methods for continuous control that adapt diffusion models from generative modeling to policy learning. Expert actions are perturbed with Gaussian noise during training, and a denoising network ϵ_θ is optimized to recover the original actions conditioned on the state. At inference, the learned reverse process iteratively refines noisy samples to generate actions, yielding a stochastic policy $p_\theta(a_t \mid s_t)$. Formally, let $\tilde{s}_t \in \mathbb{R}^d$ denote the observation at timestep t and let $a_t = \Delta p_t \in \mathbb{R}^6$ denote the end-effector displacement action. A skill policy π_i defines a conditional distribution $p_\theta(a_t \mid \tilde{s}_t)$ over displacement actions. We model this distribution using a diffusion policy (Chi et al. 2023), which parameterizes the reverse diffusion process as a denoising network ϵ_θ and generates actions by iteratively refining samples from a Gaussian prior:

$$\begin{aligned} a_t^{(0)} &\sim \mathcal{N}(0, I), \\ a_t^{(k-1)} &= \frac{1}{\sqrt{\alpha_k}}\left(a_t^{(k)} - \frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}}\epsilon_\theta\left(a_t^{(k)}, \tilde{s}_t, k\right)\right) \\ &\quad + \sqrt{1-\alpha_k}\,\epsilon, \end{aligned} \tag{1}$$

where $\{\alpha_k\}$ is the noise schedule and $\epsilon \sim \mathcal{N}(0, I)$. The diffusion formulation is particularly suited to control reference imitation because it captures multi-modal action distributions without mode averaging, a critical property when a skill can be initiated from several valid approach configurations.

Following the options framework (Sutton et al. 1999), each operator $o_i \in \mathcal{O}$ can be decomposed into J_i sequential action-step sub-policies $\{\pi_{i,1}, \ldots, \pi_{i,J_i}\}$, each active until a termination condition $\beta_{i,j} : \mathcal{H}_t \to [0, 1]$ triggers. The full operator policy is a learned finite automaton over sub-policies:

$$\pi_i = \langle \pi_{i,1}, \beta_{i,1}, \pi_{i,2}, \beta_{i,2}, \ldots, \pi_{i,J_i}, \beta_{i,J_i} \rangle, \tag{2}$$

where execution of o_i terminates when β_{i,J_i} triggers, signaling that the symbolic effects ω_i have been achieved in the continuous world.

3.3 Neuro-symbolic Models

Neuro-symbolic models combine symbolic reasoning with neural control. A planner solves a STRIPS task $T = \langle \mathcal{E}, \mathcal{F}, \mathcal{O}, s_0, s_g \rangle$ to produce a plan $\mathcal{P} = [o_1, \ldots, o_{|\mathcal{P}|}]$, where each operator o_i is refined into a neural skill $\pi_i \in \Pi$. Each skill π_i interacts with the environment to realize the operator's effects ω_i, transitioning the system from a state

s to a new state s'. This layered approach enables flexible execution in continuous spaces while maintaining high-level task abstraction.

3.4 Control Reference Level Policies

A key design choice in hierarchical robot learning is the *level of abstraction* at which imitation learning is performed. Raw actuator signals, such as joint torques or motor currents, are noisy, hardware-specific, and sensitive to the mechanical properties of the platform. Instead, we operate at the *control reference level*: policies output desired end-effector displacements $\Delta p \in \mathbb{R}^6$ in Cartesian space, which are tracked by a low-level controller (e.g., a Cartesian impedance or velocity controller) running at higher frequency. This separation between task-space imitation and joint-space tracking is consistent with classical robot control theory (Siciliano et al. 2009) and yields a smoother, less noisy imitation target that is transferable across platforms sharing similar kinematic structure [1].

3.5 Foundation Models

Foundation models (Bommasani et al. 2021) are large-scale neural networks pretrained on vast corpora of multimodal data, including images, video, and text, that acquire broad world knowledge and demonstrate strong zero-shot generalization across a wide variety of downstream tasks. In robotics, two families of foundation models are of particular relevance to our framework.

Vision-Language Models (VLMs). A VLM is parameterized as $f_\theta : (\mathcal{I}, \mathcal{Q}) \to \mathcal{R}$, where $\mathcal{I}$ denotes an image (or set of images), $\mathcal{Q}$ a natural-language query, and $\mathcal{R}$ a natural-language (or structured) response. Through pretraining on internet-scale image–text pairs, VLMs encode rich semantic knowledge about object categories, spatial relations, and action concepts. In the robotics context, VLMs have been used for task planning from human demonstrations (Wake et al. 2024), labeling of demonstration datasets (Shridhar et al. 2022), and zero-shot skill classification (Yuan et al. 2023). These models are particularly attractive for our framework because they require no task-specific fine-tuning for scene understanding: given a pair of pre- and post-skill images and a human-provided vocabulary of skill labels, a VLM can identify which skill was executed in most cases, substantially reducing the need for manual annotation. For ambiguous cases where the VLM confidence falls below a threshold, a human fallback mechanism ensures label correctness.

Open-Vocabulary Object Detectors. Object detection models such as OWLv2 (Minderer et al. 2024) are distinct from VLMs: rather than producing a global response to a query, they localize object instances as bounding boxes given a text prompt, without retraining on new categories. Formally, OWLv2 computes $\mathrm{OWL}(I, p) = \{(b_i, s_i)\}$, where I is an image, p a text prompt, b_i a predicted bounding box, and s_i the associated confidence score. In our framework, three distinct vision models serve separate purposes: a VLM for skill annotation and state equivalence judgment, OWLv2 for open-vocabulary object detection, and on platforms requiring

[1] Note that such controls could also be learned separately as primitive-level policies.

embedded inference, a compact YOLOv8 (Jocher et al. 2023) student distilled from OWLv2.

4 Problem Definition

4.1 Setting and Assumptions

We consider a robot operating in a semi-structured environment containing a set of manipulable objects $\mathcal{E} = \{e_1, \ldots, e_N\}$. The robot observes the world through a continuous observation space $\tilde{\mathcal{S}} \subseteq \mathbb{R}^d$, comprising RGB images from cameras and proprioceptive readings, and acts through a continuous action space $\mathcal{A} \subseteq \mathbb{R}^m$ of end-effector displacement commands. The true underlying world state admits a symbolic abstraction: there exists a discrete state space $\mathcal{S}$ defined over a set of boolean predicates $\mathcal{F}$ grounded on $\mathcal{E}$, and a set of operators $\mathcal{O}$ whose application transitions the world between symbolic states. Neither $\mathcal{F}$ nor $\mathcal{O}$ are known a priori; they must be inferred from data.

4.2 Demonstration Data

The learner is provided with a small demonstration dataset $\mathcal{D} = \{\tau^{(1)}, \ldots, \tau^{(n)}\}$, where each trajectory $\tau^{(k)} = \{(\tilde{s}_t, a_t)\}_{t=0}^{T_k}$ is a sequence of continuous observation-action pairs generated either by a human operator or by automated data augmentation (Section 5.2), performing a single skill. The total number of demonstrations per skill is assumed to be small: $|\mathcal{D}_{o_i}| \in [1, 30]$ for each operator o_i. No symbolic annotation is provided alongside the demonstrations: there are no predicate labels, no state segmentation markers, and no specification of planning operators. The only input is a semantic human-provided vocabulary of skill names $\Lambda = \{l_1, \ldots, l_K\}$, consistent with the level of supervision that would be required to finetune a VLA model. These labels are not assigned to trajectories.

4.3 Long-Horizon Task Execution

At execution time, the user specifies a task as a pair (s_0, s_g), where $s_0 \in \mathcal{S}$ is the initial symbolic state and $s_g \in \mathcal{S}$ is the desired goal state. Task specification can be done visually, by selecting snapshots mapped to (s_0, s_g). The system must produce a closed-loop execution that drives the robot from s_0 to s_g by sequencing learned skill policies. Success requires both correct high-level planning—selecting the right operator sequence—and reliable low-level execution of each skill under real-world perceptual noise.

4.4 Objective

The goal is to learn, from $\mathcal{D}$ and vocabulary Λ alone, a system $\Phi = (\sigma, \{\pi_i\}_{i=1}^{|\mathcal{O}|})$ consisting of a symbolic planning domain σ and a set of neural skill policies, such that Φ solves long-horizon tasks (s_0, s_g) at execution time with high success rate, generalizing across all three levels defined above, and without any further human supervision or domain engineering. Another central objective of our framework is generalization

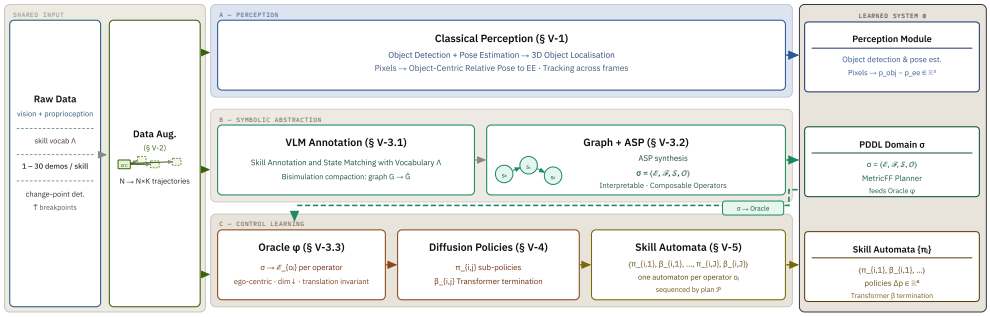


Fig. 2: **Training Pipeline.** Two shared input columns feed three processing lanes, each producing one component of the learned system Φ (right). **Shared input:** Raw demonstrations (vision, proprioception, vocabulary Λ, change-point breakpoints $\hat{T}$) are augmented by projecting waypoint skeletons onto all scene objects ($N \rightarrow N \times K$ trajectories). Augmented data feeds all three lanes. **Lane A (object detection):** OWLv2 distills a compact YOLOv8 student online. Detections are lifted to 3D and combined with proprioception to form ego-centric trajectories $\tau = \{p_{\text{obj}} - p_{\text{ee}}, a_t\}$, tracked across time. Delivers the *YOLOv8 Detector*. **Lane B (symbolic abstraction):** A VLM classifies each skill segment by comparing before/after scene images against Λ, building graph G that an ASP solver converts into PDDL domain $\sigma = \langle \mathcal{E}, \mathcal{F}, \mathcal{S}, \mathcal{O} \rangle$. Delivers *PDDL Domain* σ, and feeds σ back (dashed) to Lane C. **Lane C (control learning—sequential after Lane B):** The Oracle reads σ to construct minimal, ego-centric observations $\tilde{s}_t^{o_i}$ per operator. Diffusion sub-policies with Transformer termination predictors $\beta_{i,j}$ are assembled into per-operator skill automata. Delivers *Skill Automata* $\{\pi_i\}$.

beyond the specific object configurations seen during training. We distinguish three levels of generalization, each handled by a distinct component of our pipeline: **Intra-task generalization** refers to the ability to execute a known operator sequence when objects appear at positions not seen during training. **Inter-task generalization** refers to the ability to solve tasks with goal states s_g that require operator sequences not present in any single demonstration. **Cross-object generalization** refers to the ability to apply a known skill to object instances not present during training.

5 Neuro-symbolic Model

We utilize prior work demonstrating that neuro-symbolic models can be learned from few demonstrations (Lorang et al. 2025a) to construct our neuro-symbolic model (NSM) which combines PDDL-based symbolic planning with low-level controls. Figure 2 provides an overview of this approach. When performing a task, the model first detects the object position, then generates a high-level plan using the symbolic planner, and executes each step by using a trained diffusion policy. All diffusion policies operate on relative position observations, specifically object position expressed with respect to the end effector, and output end-effector displacement actions. The framework only receives images and proprioceptive information as input during execution.

Both the high-level planning domain and the low-level controls are learned directly from demonstrations. Like VLAs, the NSM learns to detect, plan, and act entirely from observational data, without task-specific hand-crafted controllers or manually specified symbolic domains.

5.1 Perception

5.1.1 From Pixels to Pose

Prior work (Lorang et al. 2025a) used object-based input to the framework, leaving a "perception gap" to be addressed. The proposed framework requires any perception stack to expose a single, platform-agnostic interface for object interaction: a 6-DoF pose estimate per tracked entity expressed in a frame relative to the sensor. This interface can be realized by any combination of sensors and algorithms appropriate for the deployment platform. As a concrete instantiation, we reuse the perception modules from the ADAPT system (Huemer et al. 2025), which fuses stereo-camera detections, wheel-encoder odometry, RTK-GNSS, and LiDAR measurements across three tightly coupled stages described below.

Pallet detection and 6-DoF pose estimation

A convolutional encoder-decoder network (Beleznai et al. 2025) is trained entirely on synthetic stereo image pairs to detect pallets via a part-constellation model: keypoints and edge segments vote probabilistically for pallet centers, and a depth-informed PnP solver lifts the 2D detections into a 6-DoF camera-frame pose estimate $\hat{T}_e^{\text{cam}} = (\hat{R}_e, \hat{w}_e) \in SE(3)$ for each pallet instance e.

Joint localization and pallet mapping

Raw detections $\hat{T}_e^{\text{cam}}$ are fused with wheel-encoder odometry and RTK-GNSS measurements in an iSAM2 factor graph that jointly optimizes the vehicle pose sequence $\{X_t\}$ and the global pallet poses $\{P_e\}$ in real time. Pallet detections enter the graph as binary pose factors between X_t and P_e, with diagonal Gaussian covariance whose translational and rotational uncertainties grow linearly with sensor range to reflect the distance-dependent degradation of stereo accuracy. Mahalanobis-distance data association (Mahalanobis 2018) links detections across views, while pallet bookkeeping maintains object permanence when pallets leave the sensor frustum, yielding a centimeter-accurate, incrementally updated map of all pallet poses in the mission area.

Truck loading edge detection

Prior to each unloading cycle, the target truck platform is localized once from LiDAR point clouds aggregated over time. After height filtering and edge-candidate classification, specifically points whose local neighbourhood contains both a vertical and a ground-parallel normal direction, a RANSAC line fit extracts the loading edge and its front face, defining a reference frame $T^{\text{truck}} \in SE(3)$ from which configurable pallet slot targets $\{T_k^{\text{slot}}\}$ are derived for downstream placement planning.

5.1.2 Mapping Detections to Planning Objects

To establish the mapping between tracking IDs and symbolic entities, we compute relational features (e.g., spatial proximity, relative positions) for both detected objects and ground-truth entities $\mathcal{E}$, then solve a correspondence problem that maximizes relational similarity. For instance, if the symbolic state contains "on(cube1, cube2) $\wedge$ left-of(cube3, cube1)" and the tracker detects three objects with IDs {blue_cube_0, red_cube_0, green_cube_0}, we compute the same relational predicates for each detection set and match the configuration that exhibits the highest structural similarity, producing the mapping ρ(blue_cube_0) = cube1, ρ(red_cube_0) = cube2, ρ(green_cube_0) = cube3. This bijection $\rho : \mathcal{T}_t \to \mathcal{E}$ grounds symbolic references in the visual stream and resolves scene ambiguities that arise when visually similar objects occupy symmetric configurations. I.e., without relational grounding, two red cubes in a mirrored arrangement, for example, would be indistinguishable to appearance-based tracking alone. By leveraging the symbolic state s_t and its predicates, the system can determine which physical object corresponds to which symbolic entity based on their roles in the task. This ensures that when the planner generates an action $o_i(e_j)$ for a specific entity $e_j \in \mathcal{E}$, the Oracle function ϕ (Section 5.3.3) will select the correct corresponding physical object from $\mathcal{T}_t$ via $\rho^{-1}(e_j)$, allowing the low-level policy π_i to execute the manipulation on the intended target.

5.2 Real-World Data Augmentation using Controls

A central bottleneck of data-efficient imitation learning is that a single demonstration covers only one object configuration: the trajectory from start state $\tilde{s}_0$ to goal state $\tilde{s}_g$ is entangled with the specific object poses encountered in that episode, and does not generalize to novel arrangements without additional data. For manipulation skills, where the end-effector trajectory is tightly coupled to object pose, control theory offers a principled remedy: given a target waypoint or reference trajectory, a controller can exploit a precise model of the physical embodiment, its kinematic and dynamic constraints, and its sensor characteristics to drive the system toward that target reliably across varying initial conditions. The gap between the two is therefore not one of capability, but of interface: imitation learning excels at extracting *what* to do from raw experience but remains brittle to perceptual variation, while classical control is highly competent at *how* to reach a specified target but requires that target to be symbolically and spatially well-defined. Bridging this gap—translating demonstrated intent into controller-ready references that remain valid across object configurations—is precisely the challenge this section addresses.

To multiply the effective dataset size without additional human effort, we exploit the structure of the learned control system to project a single recorded demonstration onto other objects present in the scene.

5.2.1 Trajectory projection onto new objects

For a manipulation skill, let the recorded demonstration consist of K waypoints $\mathcal{W}^{\text{src}} = \{T_k^{\text{src}}\}_{k=1}^{K} \subset SE(3)$, where each $T_k^{\text{src}} = (R_k^{\text{src}}, w_k^{\text{src}})$ encodes the end-effector orientation $R_k^{\text{src}} \in SO(3)$ and position $w_k^{\text{src}} \in \mathbb{R}^3$ at step k. Given the source (src)

and target (tgt) object poses at the start and end of the manipulation segment, $T^{\text{src}}_{\text{start}}, T^{\text{src}}_{\text{end}}, T^{\text{tgt}}_{\text{start}}, T^{\text{tgt}}_{\text{end}} \in SE(3)$, the trajectory is first rigidly re-based to the target object frame via

$$\begin{aligned} T_{\text{align}} &= T^{\text{tgt}}_{\text{start}} \big(T^{\text{src}}_{\text{start}}\big)^{-1}, \\ T^{\text{align}}_k &= T_{\text{align}} \cdot T^{\text{src}}_k, \quad k = 1, \dots, K. \end{aligned} \tag{3}$$

This re-basing aligns the trajectory origin to the target configuration, but provides no guarantee that the projected end pose T^{align}_K coincides with $T^{\text{tgt}}_{\text{end}}$, since the relative transformation between start and end may differ between source and target configurations. The residual $SE(3)$ error is computed as

$$T^{\text{res}} = T^{\text{tgt}}_{\text{end}} \big(T^{\text{align}}_K\big)^{-1}, \qquad T^{\text{res}} = (R^{\text{res}}, p^{\text{res}}), \tag{4}$$

where $p^{\text{res}} \in \mathbb{R}^3$ and $R^{\text{res}} \in SO(3)$ are its translational and rotational components, respectively. To correct the trajectory smoothly, the residual is distributed along the waypoint sequence using the normalized arc-length parameter $\alpha_k \in [0, 1]$, defined as $\alpha_k = \ell_k / \ell_K$ where $\ell_k = \sum_{j=2}^{k} \| w^{\text{align}}_j - w^{\text{align}}_{j-1} \|$ is the cumulative Euclidean path length up to waypoint k. The augmented waypoints are then

$$\begin{aligned} w^{\text{aug}}_k &= w^{\text{align}}_k + \alpha_k \, p^{\text{res}}, \\ R^{\text{aug}}_k &= \text{Slerp}(\alpha_k;\, I,\, R^{\text{res}}) \cdot R^{\text{align}}_k. \end{aligned} \tag{5}$$

where $\text{Slerp}(\alpha;\, I,\, \mathbb{R}^{\text{res}})$ denotes spherical linear interpolation in $SO(3)$ from the identity I to $\mathbb{R}^{\text{res}}$, ensuring a geodesic, minimum-torque correction of orientation. The augmented waypoint set $\mathcal{W}^{\text{aug}} = \{(R^{\text{aug}}_k,\, w^{\text{aug}}_k)\}_{k=1}^{K}$ is finally passed to the motion planner, which replans a collision-free trajectory through these references. In the spirit of MimicGen (Mandlekar et al. 2023), this procedure adapts a single demonstration to new scene configurations without additional human effort; unlike MimicGen, however, our method operates entirely at the level of abstracted control references and is deployed directly on a real robot without simulation. In the limiting case $K = 2$, the projection reduces to remapping the start and goal poses to the target configuration, with the motion planner responsible for generating the connecting trajectory, as used for navigation-based skills.

5.2.2 Dual benefit for graph construction and policy learning

The augmented trajectories serve two complementary roles in the overall pipeline. First, each augmented trajectory contributes an additional edge instance to the state-transition graph G: after VLM annotation (Section 5.3.1), it increases the statistical support per edge type, enabling the ASP solver to infer a more consistent and compact PDDL domain. Second, the augmented demonstrations provide additional training pairs $(\tilde{s}_t, a_t)$ for the diffusion policy of operator o_i, directly expanding the effective dataset size.

Despite the augmented trajectories being generated by the motion planner rather than a human, training a diffusion policy on top of them remains essential. The diffusion model learns a richer multi-modal action distribution by combining

planner-generated and human-demonstrated trajectories, generalizing more robustly to positional perturbations at execution time (Chi et al. 2023). Furthermore, the diffusion policy operates in closed-loop over relative observations at inference time, whereas the planner is open-loop with respect to the object detector; the imitation learning layer therefore absorbs residual localization errors and environmental perturbations that the planner cannot anticipate.

5.3 Automated Symbolic Abstractions for Reasoning.

5.3.1 Annotating Trajectories with Foundation Models

Each raw demonstration trajectory $\tau \in \mathcal{D}$ represents a continuous segment of robot execution during which a single skill is performed. To build the state-transition graph $G = \langle V, E, L \rangle$ required by the ASP-based symbolic abstraction stage, we must (i) identify the high-level state n before and n' after each skill execution, and (ii) assign a skill label l to each directed edge (n, l, n'). Our approach replaces expert symbolic annotation with a VLM-driven pipeline, requiring only a small human-provided vocabulary $\Lambda = \{l_1, \ldots, l_K\}$ of skill names, e.g., `unload pallet`, `load pallet` and `navigate` for the forklift domain, which are at the same level of supervision as a prompt given to a VLA to describe a task.

State representation

Each node $n \in V$ corresponds to a visual scene snapshot, concretely an image I captured at the transition between two consecutive skill executions. Nodes are identified up to bisimulation equivalence: two snapshots n and n' are merged if a VLM judges the depicted scene configurations to be semantically equivalent given short domain specific hints (e.g., "The definition of a state is primarily set by the number of pallets in the scene, the agent location and whether a pallet is currently loaded"). Formally, we query the VLM with a prompt of the form:

> *"You are an expert forklift operator assessing whether two snapshots represent the SAME high-level world state. You are given: Image A – the snapshot for node A (video: {node_a.video_id}). Image B – the snapshot for node B (video: {node_b.video_id}) Focus on: {hints}, ignore everything else. Minor pixel-level differences are acceptable — judge at the TASK level."*

together with the image pair. If the VLM returns a positive equivalence judgment, the corresponding graph nodes are merged into a single abstract state. This bisimulation compaction (Lorang et al. 2025a) reduces the number of distinct nodes in G and increases the number of skill instances available per graph edge, improving the quality of the PDDL domain subsequently inferred by the ASP solver.

Skill classification

To label each graph edge, we extract uniformly distributed frames from the demonstration segment τ, and issue a classification prompt to the VLM:

> *"You are an expert forklift operator analyzing a short manipulation sequence. You are given: 10 video frames sampled uniformly across the clip. The corresponding control trajectory. Trajectory summary: {action_trajectory} TASK: 1. Identify the single dominant*

manipulation skill performed across the entire sequence. 2. Choose the most precise label possible among the established labels: $\{l_1, \ldots, l_K\}$? Respond with the skill name only."

The VLM responds with a label $\hat{l} \in \Lambda$, which is assigned to the graph edge $(n, \hat{l}, n')$. This approach is inspired by recent work on VLM-based skill recognition and task planning from demonstration videos (Wake et al. 2024; Huang et al. 2024), and eliminates the manual trajectory labeling that most prior neuro-symbolic approaches implicitly require. In addition to such prompts, we query the VLM to output a confidence metric $c(\tau)$. When $c(\tau) < \tau_c$, the trajectory is flagged and routed to a human annotator. This mechanism decouples the scalability of the pipeline from VLM reliability: in easy, visually distinct tasks the system operates fully autonomously, while in ambiguous cases it requests the minimal human intervention necessary to preserve label correctness. In practice, we find that the fraction of flagged trajectories is small, confirming that VLMs provide reliable zero-shot skill classification when pre- and post-skill images exhibit clear visual differences.

5.3.2 Symbolic Abstraction

From raw demonstration trajectories $\mathcal{D}$, we extract node transitions $\tau^{node} = (n, l, n')$, where n and n' are high-level states and l is the VLM-assigned label, for instance `stack`. The nodes all correspond to higher-level states where n and n' correspond to the beginning and the end of the skill demonstration, respectively. These transitions form a graph $G = \langle V, E, L \rangle$ whose nodes represent abstract states and edges represent skills. To compact the structure, we compute a minimal bisimulation $\bar{G}$, removing redundant states while preserving equivalence. Using an ASP-based solver (Bonet and Geffner 2020; Rodriguez et al. 2021), we infer a symbolic domain $\sigma = \langle \mathcal{E}, \mathcal{F}, \mathcal{S}, \mathcal{O} \rangle$ in PDDL form. This yields a symbolic abstraction of the demonstrations suitable for classical planning, as well as a complete description of every node (state) in the graph.

5.3.3 Oracle-Guided Observation Filtering

A key mechanism enabling sample efficiency in our framework is the *Oracle* function ϕ which filters the full scene observation down to only the information relevant to the operator currently being executed. Formally, given the PDDL domain $\sigma = \langle \mathcal{E}, \mathcal{F}, \mathcal{S}, \mathcal{O} \rangle$ and the symbolic plan $\mathcal{P} = [o_1, \ldots, o_{|\mathcal{P}|}]$, the Oracle extracts from each operator o_i the minimal entity set:

$$\mathcal{E}_{o_i} = \left\{ e \in \mathcal{E} \;\middle|\; e \text{ appears in } \psi_{o_i} \cup \omega_{o_i} \right\}, \tag{6}$$

where ψ_{o_i} and ω_{o_i} are the preconditions and effects of o_i, respectively. At execution timestep t, the tracking system provides 3D position estimates $\mathcal{P}_t$ for all detected objects. The Oracle resolves symbolic entities to physical detections via the bijection $\rho : \mathcal{T}_t \to \mathcal{E}$ and constructs a filtered, ego-centric observation:

$$\tilde{s}_t^{o_i} = \left\{ p_e^{\text{rel}} = p_{\rho^{-1}(e)} - p_{\text{ee}} \;\middle|\; e \in \mathcal{E}_{o_i} \right\}, \tag{7}$$

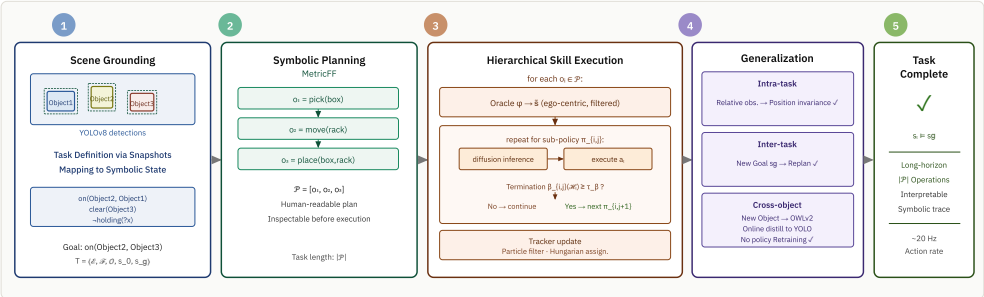


Fig. 3: *Execution Pipeline.* At execution time, the system proceeds through five stages. **(1) Scene grounding:** YOLOv8 detects task-relevant objects, 3D positions are estimated, and relational predicates are evaluated to construct the initial symbolic state s_0. The user specifies a goal s_g as a partial state. **(2) Symbolic planning:** MetricFF solves the PDDL instance $\mathcal{T} = \langle \mathcal{E}, \mathcal{F}, \mathcal{O}, s_0, s_g \rangle$ in milliseconds, producing a human-readable, inspectable plan $\mathcal{P}$. **(3) Hierarchical skill execution:** For each operator o_i, the Oracle ϕ resolves symbolic entity bindings to physical detections and constructs the ego-centric observation. Diffusion sub-policies execute closed-loop at $\sim$20 Hz; a Transformer termination predictor $\beta_{i,j}$ gates transitions between action steps. The tracker updates the symbolic state after each operator completes. **(4) Generalization:** Three levels of generalization are handled by distinct mechanisms—relative observations for intra-task generalization, replanning for inter-task generalization, and on-the-fly OWLv2 distillation for cross-object transfer. **(5) Termination:** Execution terminates when $s_t \models s_g$.

where $p_{\text{ee}} \in \mathbb{R}^3$ is the current end-effector position. For unary operators (i.e., $|\mathcal{E}_{o_i}| = 1$), the Oracle additionally computes the full relative 6-DoF transform between the end effector and the target object, providing a compact geometric signal that encodes both position and orientation offsets. The observation $\tilde{s}_t^{o_i}$ is then passed as input to the corresponding diffusion policy π_i.

This design has two important consequences for generalization. First, expressing object positions relative to the end effector makes the observation space translation-invariant: a policy trained on one absolute scene configuration transfers directly to new configurations as long as the relative geometry is similar, which is guaranteed when operating on the filtered entity set $\mathcal{E}_{o_i}$. Second, the dimensionality of $\tilde{s}_t^{o_i}$ scales with $|\mathcal{E}_{o_i}|$ rather than with the total number of scene objects $|\mathcal{E}|$, dramatically reducing the input space and the amount of demonstration data required to train a reliable policy for each operator.

5.4 Neural Plasticity for Skill Learning

5.4.1 Training Controls

Given the symbolic domain σ and the demonstration dataset $\mathcal{D}$, we instantiate and train the operator policy automaton of Equation 2 for each operator $o_i \in \mathcal{O}$ as follows.

Sub-policy decomposition

The change-point breakpoints $\hat{\mathcal{T}}$ extracted from each operator demonstration segment the trajectory into J_i phases corresponding to behaviorally distinct stages of the manipulation (e.g., approach, contact, retraction). Each phase defines the training data for one sub-policy $\pi_{i,j}$: the demonstration segments between consecutive breakpoints $[t_{j-1}, t_j]$ provide the observation-action pairs $\{(\tilde{s}_t^{o_i}, a_t)\}$ used for diffusion policy training. The same change-point breakpoints $\hat{\mathcal{T}} = \{t_1, \ldots, t_K\}$ additionally provide the supervision signal for termination learning. For each sub-policy $\pi_{i,j}$, the positive class (termination) is defined by a window of δ timesteps centered on each detected breakpoint t_k, and the negative class by all remaining timesteps. This creates a natural and annotation-free labeling of termination events directly from the motion signal, with no additional human supervision required.

Diffusion policy training

Each sub-policy $\pi_{i,j}$ is trained by minimizing the standard diffusion denoising objective over its assigned demonstration segments:

$$\mathcal{L}_\pi = \mathbb{E}_{(\tilde{s}_t, a_t),\, k,\, \epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_k}\, a_t + \sqrt{1-\bar{\alpha}_k}\, \epsilon,\ \tilde{s}_t^{o_i},\ k\right)\right\|^2\right], \tag{8}$$

where $k \sim \mathcal{U}(1, K)$ is a uniformly sampled diffusion timestep and $\epsilon \sim \mathcal{N}(0, I)$. Training data for each sub-policy is augmented with the projected demonstrations generated by the data augmentation procedure (Section 5.2), multiplying the effective dataset size without additional human effort.

5.4.2 Termination Condition Learning

Each skill operator o_i is decomposed into a sequence of action-step sub-policies $\{\pi_{i,1}, \ldots, \pi_{i,J_i}\}$, inspired by the options framework (Sutton et al. 1999). Each sub-policy $\pi_{i,j}$ is associated with a learned termination condition $\beta_{i,j} : \mathcal{H}_t \to [0, 1]$, which predicts the probability that the current action step has been completed and execution should transition to $\pi_{i,j+1}$. Termination is triggered when $\beta_{i,j}(\mathcal{H}_t) \geq \tau_\beta$ for a threshold τ_β.

Transformer termination predictor

Termination prediction requires reasoning over a temporal context window rather than a single observation snapshot, since the completion of a contact-rich action step (e.g., a grasp closing or a pallet insertion) manifests itself as a pattern in recent observations rather than an instantaneous state change. We, therefore, model $\beta_{i,j}$ as a Transformer (Vaswani et al. 2017) sequence classifier operating over a sliding history window $\mathcal{H}_t = (\tilde{s}_{t-W}, \ldots, \tilde{s}_t) \in \mathbb{R}^{W \times d}$ of the last W filtered observations.

The model first projects observations into a latent space of dimension d_h and adds a learned positional encoding:

$$\mathbf{H}^{(0)} = \mathbf{X}\mathbf{W}_{\text{in}} + \mathbf{P}, \quad \mathbf{X} \in \mathbb{R}^{W \times d},\ \mathbf{W}_{\text{in}} \in \mathbb{R}^{d \times d_h},\ \mathbf{P} \in \mathbb{R}^{W \times d_h}, \tag{9}$$

where $\mathbf{P}$ is a learned parameter matrix. The projected sequence is processed by an L-layer Transformer encoder with N_h attention heads and feedforward dimension $4d_h$:

$$\mathbf{H}^{(\ell)} = \text{TransformerLayer}\left(\mathbf{H}^{(\ell-1)}\right), \quad \ell = 1, \ldots, L. \tag{10}$$

The representation of the final token $\mathbf{h}_W = \mathbf{H}^{(L)}_{W,:} \in \mathbb{R}^{d_h}$ aggregates the full causal context and is passed through a two-layer MLP prediction head:

$$\beta_{i,j}(\mathcal{H}_t) = \sigma(\mathbf{W}_2 \,\text{ReLU}(\mathbf{W}_1 \mathbf{h}_W + \mathbf{b}_1) + \mathbf{b}_2) \in [0, 1], \tag{11}$$

where σ denotes the sigmoid activation. The model is trained with binary cross-entropy loss:

$$\mathcal{L}_\beta = -\frac{1}{T} \sum_{t=1}^{T} \left[y_t \log \beta_{i,j}(\mathcal{H}_t) + (1 - y_t) \log\big(1 - \beta_{i,j}(\mathcal{H}_t)\big) \right], \tag{12}$$

where $y_t = 1$ if timestep t falls within the δ-window of a breakpoint and $y_t = 0$ otherwise. Class imbalance, since termination events are sparse relative to execution timesteps, is addressed by positive-class reweighting with weight $\gamma = T/(2|\hat{\mathcal{T}}|\delta)$.

The Transformer architecture is preferred over recurrent alternatives (e.g., LSTM) for this task because the self-attention mechanism can directly relate the current observation to any prior timestep in the window, capturing long-range dependencies in contact dynamics without the vanishing gradient limitations of sequential state updates (Vaswani et al. 2017). In practice, we use $L = 3$ layers, $N_h = 4$ attention heads, $d_h = 128$, and a history window of $W = 50$ timesteps.

5.5 Execution of the Framework

At execution time, the system operates as a closed-loop hierarchical executor that interleaves symbolic planning with continuous skill execution. Figure 3 illustrates the full execution pipeline. We describe each stage in turn.

5.5.1 Task specification and grounding

A task is defined by an initial state $s_0 \in \mathcal{S}$ and a goal state $s_g \in \mathcal{S}$ expressed over the predicate vocabulary $\mathcal{F}$ of the learned domain σ. In practice, the user selects a pair of snapshots from the set of symbolic states observed during training; since each snapshot is directly associated with a set of grounded predicates, planning is immediate upon this selection. If no suitable snapshot exists for s_0 or s_g, the user must manually compose the state by specifying the relevant predicates from $\mathcal{F}$, which requires interpreting the ASP-synthesized PDDL domain σ directly. Together, these form the PDDL planning instance:

$$T = \langle \mathcal{E}, \mathcal{F}, \mathcal{O}, s_0, s_g \rangle. \tag{13}$$

Algorithm 1 Framework Execution at execution time

Require: Planning domain σ, policies $\{\pi_i\}$, initial state s_0, goal s_g

Compute plan $\mathcal{P} = [o_1, \ldots, o_{|\mathcal{P}|}]$ via MetricFF on $T = \langle \mathcal{E}, \mathcal{F}, \mathcal{O}, s_0, s_g \rangle$
for each operator $o_i \in \mathcal{P}$ **do**
  Resolve entities: $\mathcal{E}_{o_i} \leftarrow \phi(o_i, \mathcal{T}_t)$ ▷ Oracle filtering
  for each sub-policy $\pi_{i,j}$, $j = 1, \ldots, J_i$ **do**
    repeat
      Observe $\tilde{s}_t^{o_i} \leftarrow$ ego-centric filtered observation
      Sample $a_t \sim p_\theta(\cdot \mid \tilde{s}_t^{o_i})$ ▷ Diffusion policy
      Execute a_t on robot
      Update tracker $\mathcal{T}_{t+1}$, $t \leftarrow t + 1$
    until $\beta_{i,j}(\mathcal{H}_t) \geq \tau_\beta$ ▷ Termination condition
  end for
  Apply effects: $s_t \leftarrow s_t \oplus \omega_i$
end for
return success if $s_t \models s_g$, failure otherwise

5.5.2 Symbolic planning

The planning instance T is passed to MetricFF (Hoffmann 2003), a classical forward-chaining heuristic planner, which searches for a sequence of grounded operators $\mathcal{P} = [o_1, \ldots, o_{|\mathcal{P}|}]$ that transitions s_0 to a state satisfying s_g. Each grounded operator $\hat{o}_i = o_i(e_i^1, \ldots, e_i^{k_i})$ binds specific entities from $\mathcal{E}$ to the operator parameters. Planning operates entirely in the discrete symbolic space and completes in milliseconds for the task horizons considered, making it negligible in the overall execution budget. The resulting plan $\mathcal{P}$ is human-readable and can be inspected or overridden by an operator before execution begins, providing a natural point of human oversight.

5.5.3 Hierarchical skill execution

Execution proceeds by iterating over the plan $\mathcal{P}$ and invoking each operator policy π_i in sequence. For each operator o_i, the Oracle ϕ resolves the bound entities $\{e_i^1, \ldots, e_i^{k_i}\}$ to their current physical detections via ρ^{-1} and constructs the filtered ego-centric observation $\tilde{s}_t^{o_i}$ (Equation 7). The operator policy automaton $\pi_i = \langle \pi_{i,1}, \beta_{i,1}, \ldots, \pi_{i,J_i}, \beta_{i,J_i} \rangle$ then executes its sub-policies sequentially: at each timestep t, the active sub-policy $\pi_{i,j}$ samples a Cartesian displacement action via the reverse diffusion process (Equation 1), which is forwarded to the low-level Cartesian controller. The Transformer termination predictor $\beta_{i,j}$ evaluates the observation history $\mathcal{H}_t$ at every timestep; when $\beta_{i,j}(\mathcal{H}_t) \geq \tau_\beta$, execution advances to $\pi_{i,j+1}$. When the final sub-policy π_{i,J_i} terminates, operator o_i is considered complete and the tracker updates the symbolic state s_t by applying the effects ω_i. The full execution loop is summarized in Algorithm 1.

5.5.4 Generalization at execution time

The hierarchical structure of Algorithm 1 provides generalization across all three levels defined in Section 4. Intra-task generalization is handled implicitly: because $\tilde{s}_t^{o_i}$ is

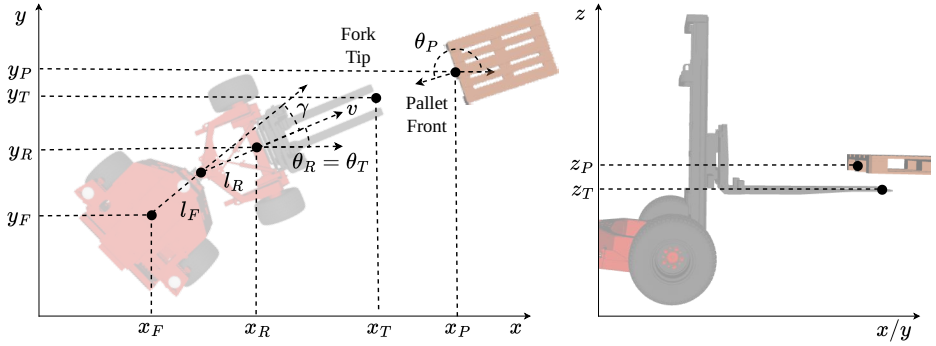


Fig. 4: **Pallet alignment problem with respect to ADAPT kinematics.** Relevant states include: rear body pose $\mathbf{q}_R = [x_R, y_R, \theta_R]^{\mathrm{T}}$, fork tip pose $\mathbf{q}_T = [x_T, y_T, \theta_T, z_T]^{\mathrm{T}}$, and target pallet pose $\mathbf{q}_P = [x_P, y_P, \theta_P, z_P]^{\mathrm{T}}$, along with the articulation angle γ and control inputs v and $\dot{\gamma}$.

recomputed at every timestep from live detector estimates, the policy always operates on the current relative geometry rather than any memorized absolute position, naturally accommodating object displacements due to prior skill executions or environmental perturbations. Inter-task generalization is achieved by replanning: for a new goal s'_g not present in any training episode, MetricFF composes the available operators into a new sequence $\mathcal{P}'$ without requiring any additional demonstrations, as long as each individual operator in $\mathcal{P}'$ has been learned. Cross-object generalization follows naturally from PDDL operators being lifted: at execution time, any object of the correct type can be substituted into an operator, regardless of whether that specific instance appeared in training demonstrations, provided the perception module can supply reliable pose estimates for it [2].

6 Evaluations

We evaluate our Neuro-Symbolic model on two robots in different settings which we will describe in the following.

6.1 Robotic Systems

6.1.1 Autonomous Outdoor Forklift

The first evaluation system is the ADAPT (Autonomous Dynamic All-terrain Pallet Transporter) autonomous outdoor forklift for pallet handling in unstructured environments (Huemer et al. 2025) (see Figure 1). The vehicle is hydraulically actuated and consists of a two-part articulated chassis where the front body houses the motor and

[2]Integrating a completely new object within the planning domain requires a semantic name label being detected by an open-world detector and an object-type classification for planning.

the rear body carries the fork lift mechanism. The system bundles five core capabilities required for fully autonomous pallet transport, which are all explained in detail in (Huemer et al. 2025). We provide a brief overview for completeness:

- **Navigation:** Collision-free, bi-directional path planning with a Hybrid A* planner adapted for articulated vehicles, combined with path tracking.
- **Localization and Mapping:** A factor-graph-based joint vehicle localization and pallet mapping framework providing centimeter-accurate pose estimates.
- **Pallet Detection:** A synthetically trained, geometry-based neural network pipeline for pallet detection and 6-DOF pose estimation.
- **Manipulation:** Precise pallet pick-up and placement via cascaded vehicle base and fork actuation control loops.
- **Obstacle Avoidance:** Real-time LiDAR-based terrain mapping and dynamic obstacle detection.

The aspect most relevant to this work is the task of picking up and placing pallets, where vehicle base control and fork positioning control play a crucial role, as they are actively used for data augmentation. Both are illustrated in Fig. 4 and explained as follows.

Vehicle Base Control

ADAPT's chassis consists of two rigid bodies connected by a center articulation joint. The rear body pose in the 2D plane is $\mathbf{q}_R = [x_R, y_R, \theta_R]^{\mathrm{T}}$, and the articulation angle between the two bodies is γ. The kinematic model relating the state derivatives to the control inputs velocity v and steering rate $\dot{\gamma}$ is:

$$\begin{bmatrix} \dot{\mathbf{q}}_R \\ \dot{\gamma} \end{bmatrix} = \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta}_R \\ \dot{\gamma} \end{bmatrix} = \begin{bmatrix} \cos(\theta_R) & 0 \\ \sin(\theta_R) & 0 \\ \frac{\sin(\gamma)}{l_R \cos(\gamma) + l_F} & \frac{l_F}{l_R \cos(\gamma) + l_F} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \dot{\gamma} \end{bmatrix}, \tag{14}$$

where l_F and l_R are the distances from the articulation point to the front and rear axles, respectively. The vehicle base is controlled by tracking the pallet pose $\mathbf{q}^d = [x_P, y_P, \theta_P]^{\mathrm{T}}$ as reference, so that the vehicle drives up to and aligns with the pallet front. The pose error expressed in the vehicle body-fixed frame is:

$$\begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} = \begin{bmatrix} \cos(\theta_P) & \sin(\theta_P) & 0 \\ -\sin(\theta_P) & \cos(\theta_P) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_R - x_P \\ y_R - y_P \\ \theta_R - \theta_P \end{bmatrix}, \tag{15}$$

comprising the longitudinal error e_x, lateral error e_y, and heading error e_θ. Minimising e_y and e_θ yields the desired steering rate

$$\dot{\gamma}^d = -\frac{K_1 v (l_F + l_R)}{l_R} e_y - \frac{K_2 (l_F + l_R)}{l_R} e_\theta - \frac{v}{l_R} \gamma, \tag{16}$$

derived via Lyapunov-based analysis, where K_1 and K_2 are control gains. The desired velocity is

$$v^d = \max\big(v_R - k_v(\dot{\gamma}^d)^2,\ v_{\min}\big)\,, \tag{17}$$

reducing speed proportionally to curvature to maintain tracking accuracy, where v_R is the nominal reference velocity and $v_{\min}$ a lower speed bound.

Fork Positioning Control

Pallet pick-up requires the fork tips $\mathbf{q}_T = [x_T, y_T, \theta_T, z_T]^{\mathrm{T}}$ to be precisely aligned with the pallet front face $\mathbf{q}_P = [x_P, y_P, \theta_P, z_P]^{\mathrm{T}}$. The pallet position is transformed into the fork tip coordinate frame:

$$\begin{bmatrix} x'_P \\ y'_P \end{bmatrix} = \begin{bmatrix} \cos(e_\theta) & -\sin(e_\theta) \\ \sin(e_\theta) & \cos(e_\theta) \end{bmatrix} \begin{bmatrix} x_P - x_T \\ y_P - y_T \end{bmatrix}, \tag{18}$$

where $e_\theta = \theta_P - \theta_T$ is the heading misalignment, and x'_P, y'_P are the longitudinal and lateral distances to the pallet expressed in the fork tip frame. Based on the transformation output, the reference signals for the fork actuators are derived directly: the fork shift reference $s^d = y'_P$ drives lateral alignment, the lift mast height reference $l^d = z_P$ corrects the vertical offset, and the tilt reference β^d keeps the forks parallel to the ground. The heading error e_θ is reduced by the vehicle base control loop described above.

Computational profile

The dominant cost at runtime is diffusion policy inference, which requires K denoising steps per action; in practice we use $K = 10$, yielding action inference at approximately $20\,Hz$. Perception and planning run concurrently on the same hardware, with symbolic planning introducing negligible overhead relative to policy inference. All components run on-board on an NVIDIA L4 GPU (Nuvo-9166GC embedded IPC), confirming the system's viability for edge deployment without any cloud offloading.

6.2 Evaluation Settings

We evaluate the full framework across three distinct experimental settings designed to probe complementary aspects of the system: data efficiency and manipulation precision on the real industrial forklift, adaptive skill extension via single-demonstration transfer, and cross-platform domain independence on the Kinova Gen3 robotic arm.

6.2.1 Statistical Evaluation of Forklift Pallet Manipulation

To rigorously quantify the manipulation performance of the framework under realistic variability, we conducted a large-scale statistical evaluation of the full load-navigate-unload cycle on the ADAPT platform. Rather than fixing the initial vehicle pose relative to the pallet, we reset the forklift autonomously between trials by exploiting the onboard navigation module and the fork actuation control loops. Concretely, at

the start of each trial, a target reset pose $\mathbf{q}_{\text{reset}} = [x, y, \theta]^\top$ is sampled from a truncated Gaussian distribution

$$\mathbf{q}_{\text{reset}} \sim \mathcal{N}(\boldsymbol{\mu}_{\text{nominal}}, \Sigma_{\text{reset}}) \,\big|_{\mathbf{q} \in \mathcal{R}}, \tag{19}$$

where $\boldsymbol{\mu}_{\text{nominal}}$ is the nominal approach pose used during training, $\Sigma_{\text{reset}} = \text{diag}(\sigma_x^2, \sigma_y^2, \sigma_\theta^2)$ encodes positional and angular uncertainty, and $\mathcal{R}$ is a rectangular feasibility region that excludes poses that would result in irreparable initial collisions or sensor occlusions. We set $\sigma_x = \sigma_y = 0.4\,\text{m}$ and $\sigma_\theta = 15°$, reflecting the realistic range of positioning errors encountered in unstructured outdoor environments. The navigation module drives the vehicle autonomously to $\mathbf{q}_{\text{reset}}$ before each trial, without any manual repositioning, making the evaluation fully unsupervised.

We report results under two complementary success criteria: *strong* success requires task completion without either a strong collision (contact force exceeding a threshold) or a significant pallet displacement ($> 10\,\text{cm}$ lateral shift at deposit); *soft* success permits minor contact events and marginal pallet displacements ($< 10\,\text{cm}$), reflecting industrial-grade tolerance requirements. Figure 7 summarizes the aggregate success rates as a function of the number of demonstrations per skill. To further characterize the performance dependence on the initial pose, Figure 9 shows the distribution of successes and failures over the forklift reset region, where each episode is represented by an arrow encoding the initial $(x_{\text{init}}, y_{\text{init}}, \psi_{\text{init}})$ pose relative to the pallet.

Across the 30- and 10-demonstration settings, no strong spatial correlation is observed between initial pose and outcome, indicating that the policy generalizes robustly across the sampled reset distribution. In the 5-demonstration regime, sensitivity increases notably along the longitudinal axis x, with a coherent success region spanning approximately 50 cm, beyond which performance degrades. Failure modes are primarily attributed to policy training quality, as demonstration count is the dominant factor governing robustness. Nevertheless, we note that in some instances, perception noise such as lighting conditions adversely affects object detection and task success. We do not observe any obvious correlation between the pallet geometric pose and the agent performance.

6.2.2 Long-Horizon Task Execution from Minimal Demonstrations

To stress-test the data efficiency of the symbolic planning component, we evaluated the framework on a compound long-horizon task involving *two pallets* and *two distinct drop-off zones*, a setting that requires the planner to sequence eight high-level operators (`navigate`, `load_pallet`, `navigate`, `unload_pallet` $\times 2$ per pallet) in the correct order. Importantly, no single training demonstration ever depicted a two-pallet, two-zone task; rather the system was trained exclusively from demonstrations of individual skills, each demonstrated at most $n \in \{1, 5, 10, 30\}$ times. At evaluation time, the user specifies the task by selecting the initial and goal snapshots from the set of symbolic states observed during training; MetricFF then synthesizes the correct multi-step plan in less than one second.

This experiment directly validates *inter-task generalization*: the ability to compose learned operators into novel sequences not represented in any single demonstration



Fig. 5: Main evaluation domain: an automated forklift for managing multiple pallets in outdoor scenario. The pallets can be stored on the ground or on a truck platform. Image credit: © AIT/tm-photography.

trajectory. The system successfully completed the two-pallet task in 9/10 trials with 10 base demonstrations per skill, confirming that the symbolic planner can extrapolate compositionally well beyond the demonstrated task horizon.

6.2.3 Single-Demonstration Adaptation to a Novel Drop-Off Location

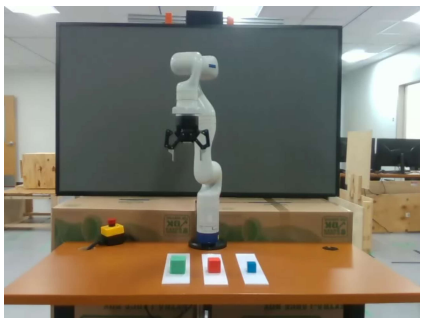
A central practical requirement in warehouse and construction-site deployments is the ability to extend a deployed system to new task configurations without full retraining. We demonstrate this by introducing a *truck-bed drop-off* location that sits $z \approx 1.7\,\mathrm{m}$ above ground, substantially outside the vertical range covered by the original training demonstrations, which were all performed at ground level.

A single demonstration of the new unload task with z height elevation (the height of the truck loading area) `unload_pallet_on_truck` skill was provided. The human also classified such skill as complementary with the existing base `unload` skill. The Oracle reuses the operator definition to identify the minimal observation space for the new skill, subtracts the dimensions already covered by existing skills, and returns only the residual, here, the relative z position between the end-effector and the drop-off location. A diffusion policy for this skill is then trained from the single demonstration. The ASP solver does not need to re-runs in such scenario, as no additional abstraction is required. A new location is simply added as one additional entity in the PDDL problem definition. At evaluation time, the planner seamlessly integrates the new operator into existing multi-step plans, for example, routing pallet transport to the truck rather than a ground zone, without any modification to the previously trained sub-policies.

This result highlights a key advantage of the neuro-symbolic architecture: because skills are compositional building blocks managed by a symbolic planner, extending the task repertoire requires adding a single leaf to the operator graph rather than retraining an end-to-end model from scratch. The new operator is available for planning immediately after the diffusion policy has been trained on the (augmented) single demonstration.



a: Kinova Kitchen



b: Kinova Stack

Fig. 6: *Validation domains:* We validate our approach on a different embodiment (Kinova robotic arm) across two domains: `Kitchen`, where the arm must organize objects on a kitchen table, and `Stacking`, where it must stack cubes in the order of their size. The architecture mirrors the forklift domain, with the perception stack replaced by OWLv2 for object detection and depth data for 3D position estimation. From a *single* human demonstration, the agent extracts waypoints, projects them onto new object configurations, and executes the corresponding controls to complete each task end-to-end. We further validate a knowledge distillation pathway—OWLv2 supervising a lightweight YOLOv8 model, and demonstrated controls supervising compact diffusion policy networks—confirming that the architecture can be deployed on resource-constrained hardware without sacrificing task performance.

6.2.4 Cross-Platform Validation on a Kinova Gen3 Manipulator

To demonstrate platform independence, we deployed the same symbolic abstraction and learning pipeline unchanged on a Kinova Gen3 7-DOF arm with a Robotiq 2F-85 gripper, across two standard benchmarks: *block stacking* and *kitchen manipulation*. The sole platform-specific adaptation is the perception stack, where pallet detection is replaced by OWLv2 detections distilled into a YOLOv8 model. The same code path, prompts, and hyperparameters used for the forklift produce valid PDDL domains for tabletop manipulation without modification, confirming that VLM-driven annotation and ASP-based domain synthesis are genuinely domain-agnostic. Both environments are evaluated over 20 goal configurations.

End-Effector Control

The arm is controlled in Cartesian space via a differential inverse kinematics controller. Given a desired end-effector pose $T^d = (R^d, w^d) \in SE(3)$, the task-space error is

$$e = \begin{bmatrix} w^d - w \\ \mathrm{Log}(R^\top R^d) \end{bmatrix} \in \mathbb{R}^6, \tag{20}$$

where $w \in \mathbb{R}^3$ is the current end-effector position, $R \in SO(3)$ its orientation, and $\mathrm{Log}(\cdot)$ the $SO(3)$ logarithmic map returning the rotation vector of the relative orientation error expressed in the current body frame. The desired joint velocities are computed via the damped least-squares pseudoinverse of the geometric Jacobian $J \in \mathbb{R}^{6\times 7}$:

$$\dot{q}^d = J^\dagger(\lambda)\, K\, e, \qquad J^\dagger(\lambda) = J^\top \left(JJ^\top + \lambda^2 I\right)^{-1}, \tag{21}$$

where $K \in \mathbb{R}^{6\times 6}$ is a diagonal proportional gain matrix and $\lambda > 0$ a damping factor ensuring numerical stability near singularities.

Grasp Execution

Grasp and release actions are triggered symbolically by the task planner and executed open-loop by the Robotiq gripper controller, with grasp width set as a function of the target object type as specified in the PDDL domain.

Perception pipeline

On the Kinova platform, we replaced the pallet-specific detector with the open-vocabulary OWLv2 model (Minderer et al. 2024), prompted at runtime with the target object names (e.g., `red cube`, `mug`, `drawer handle`). OWLv2 produces axis-aligned bounding boxes with associated confidence scores; 3D object positions are recovered by back-projecting the box centroid through the calibrated depth image from a wrist-mounted RGB-D sensor, yielding metric estimates $\hat{\mathbf{p}}_e \in \mathbb{R}^3$ for each detected entity. A Hungarian-algorithm-based multi-object tracker maintains consistent track identities across frames, resolving the correspondence between YOLOv8 detection IDs and symbolic entities $\mathcal{E}$ via the relational matching procedure described in Section 5.1.2. This fully perception-agnostic setup required no additional annotation beyond the skill vocabulary Λ.

Manipulation benchmarks

The *block stacking* benchmark presents the robot with three colored cubes in randomized configurations and requires building a target tower arrangement — a long-horizon task (up to six chained operators) demanding both intra-task generalization (unseen initial positions) and inter-task generalization (novel tower orderings not seen during training). A single `pick_and_place` demonstration was provided, augmented to 20 trajectories via the pipeline described in Section 5.2. The *kitchen manipulation* benchmark increases perceptual and geometric complexity, requiring the robot to manipulate objects with greater intra-category shape variation (pan, basket, vegetables), and the planner to sequence heterogeneous skills over a longer horizon. We reused the same controls learned on the *block stacking* tasks accross the *kitchen manipulation* tasks.

Computational profile

On the kinova arm, we use $K = 25$ diffusion inferences per action step, yielding action inference at approximately $25\,Hz$. Perception (Owlv2 or Yolov8 (Minderer et al. 2024; Jocher et al. 2023)) and planning still run concurrently on the same hardware (Nvidia RTX4090). Object detection runs asynchronously at the YOLOv8 inference

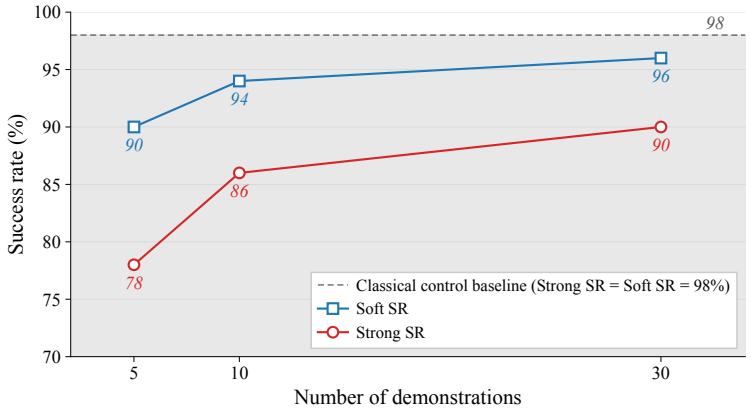


Fig. 7: Success rate of the neurosymbolic approach as a function of the number of demonstrations for real-world forklift pallet management. The task consists of loading a pallet, navigating to a target area, and unloading it under randomized relative position to the pallet. We report two criteria: Strong SR (no significant collisions or pallet displacement) and Soft SR (task completed but minor collisions or pallet pushing allowed). The dashed line indicates the classical control baseline, which achieves 98% under both criteria. Performance improves monotonically with additional demonstrations, with Soft SR approaching the baseline at 30 demonstrations.

rate ($> 30\,Hz$), decoupled from the policy inference loop via the particle filter tracker which bridges any detection latency.

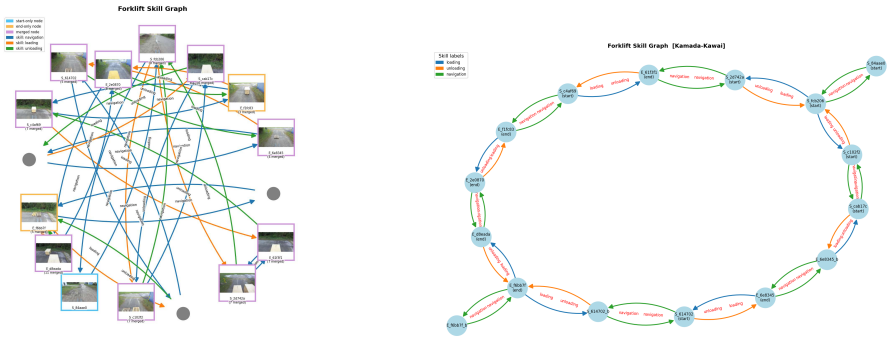
7 Results

7.1 Trajectories to Plans

7.1.1 VLM Graph Construction

The graph shown in Fig. 8a is constructed by feeding demonstration trajectories to a VLM, which clusters visually similar high-level states into shared nodes and annotates the resulting edges with the corresponding skill labels (*navigate*, *load*, *unload*). Each skill was demonstrated 5 times. The resulting graph is then reduced via bisimulation (Fig. 8b), collapsing behaviorally equivalent nodes into canonical representatives before being passed to the ASP solver.

We employ Gemini 3 Flash DeepMind (2025) as our VLM-based graph constructor. Given a pair of observations from two distinct demonstrations, the model is prompted to output a binary label (`true`/`false`) indicating whether the two observations correspond to the same abstract state, and thus should be merged into a single node. Edges between nodes are then annotated with a single-word skill label (`navigate`, `load`, or `unload`), derived from the skill identifier associated with the transition in the demonstration trajectory. The graph is built incrementally over 5 demonstrations per skill: each new state, i.e., start or end snapshot of the demonstrated skill trajectory, is compared against all existing nodes, and either merged into a matching node or



a: VLM Graph Construction Result

b: Graph bisimulation fed to the ASP Solver

Fig. 8: VLM-constructed state-transition graph with three added nodes for visual completeness (left) and its bisimulation reduction (right), used as input to the ASP solver for automated action model induction. A bigger version of both graphs is attached in Appendix. A.

instantiated as a new one, with the corresponding labelled edge added. As reported in Table 1, the VLM achieves high accuracy on edge annotation (98%) but more modest accuracy on node matching (85%), confirming that human supervision remains necessary to ensure graph correctness with current VLMs.

Table 1: VLM graph construction accuracy using Gemini 3 Flash.

Task	**Accuracy**
Node matching (state equivalence, boolean)	85%
Edge annotation (skill label, one word)	98%

In Fig. 8a, 3 nodes were manually injected by a human operator for full graph visualization, nevertheless a complete and noise-free graph is *not* a prerequisite for the pipeline to yield a functional planning domain: the ASP can, to some extent, operate over partial or imperfect graphs and still produce an executable domain (see results published by Bonet and Geffner (2020)). However, missing nodes or edges may cause the solver to generalize over fewer observed transitions, potentially resulting in action models that are correct but overly specific, requiring more parameters, additional preconditions, or failing to unify effects that a fully observed graph would have merged. The comparison between full and partial graph outputs is analyzed in the next paragraph.

7.1.2 Post-Hoc PDDL output Interpretation

The output of the ASP Solver does not come with semantics attached; a human needs to interpret the gross ASP outputs reported in Appendix. A. The following action models are the interpreted results of the ASP computation over the graph structure of the state space of the forklift domain, using either the full or partial observation graphs. The action models for the Kinova domain are reused from Lorang et al. (2025a).

Forklift Multiple Pallets Storage Domain

Full Domain Graph

MOVE: $a_1(x_1, x_2)$
Pre: (free_location x_1), (forklift_at x_2)
Eff: (not (free_location x_1)), (not (forklift_at x_2)),
(forklift_at x_1), (free_location x_2)

UNLOAD: $a_2(x_1, x_2)$
Pre: (forklift_at x_2), (loaded_pallet x_1)
Eff: (not (loaded_pallet x_1)), (free_forklift), (at x_1 x_2)

LOAD: $a_3(x_1, x_2)$
Pre: (free_forklift), (forklift_at x_2), (at x_1 x_2)
Eff: (not (free_forklift)), (not (at x_1 x_2)), (loaded_pallet x_1)

Objects: o_1, o_2, o_3, o_4 *(2 pallets × 2 locations)*

One Node, Two Edges Missing

MOVE: $a_1(x_1, x_2, x_3)$

Static: $\neg(x_1{=}x_2)$, $\neg(x_1{=}x_3)$
Pre: (at x_2 x_1), (at x_2 x_2), (connected x_3 x_1)
Eff: (not (at x_2 x_1)), (not (at x_2 x_2)),
(at x_1 x_1), (at x_1 x_2)

UNLOAD: $a_2(x_1, x_2)$

Static: $\neg(x_1{=}x_2)$
Pre: (loaded x_1), (at x_1 x_1), (at x_2 x_2), (connected x_2 x_2)
Eff: (not (loaded x_1)), (not (at x_2 x_2)),
(free_forklift), (at x_1 x_2)

LOAD: $a_3(x_1, x_2)$

Static: $\neg(x_1{=}x_2)$
Pre: (free_forklift), (at x_2 x_1), (at x_2 x_2), (connected x_1 x_1)
Eff: (not (free_forklift)), (not (at x_2 x_1)),
(loaded x_2), (at x_1 x_1)

Objects: o_1, o_2, o_3, o_4 *(2 pallets × 2 locations)*

Two Nodes, Six Edges Missing
MOVE: $a_1(x_1, x_2, x_3)$

Static: $\neg(x_1{=}x_2)$
Pre: `(at` x_1`)`, `(free` x_3`)`,
`(connected` x_1 x_1`)`, `(connected` x_2 x_2`)`, `(connected` x_3 x_1`)`
Eff: `(not (at` x_1`))`, `(at` x_2`)`
UNLOAD: $a_2(x_1, x_2)$

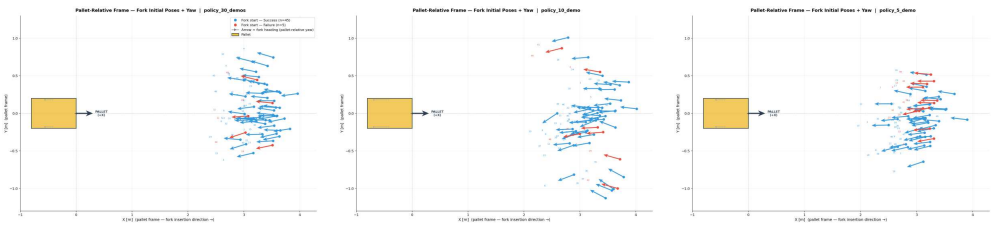
Static: $\neg(x_1{=}x_2)$
Pre: `(at` x_1`)`, `(at` x_2`)`, `(loaded` x_1`)`, `(connected` x_1 x_2`)`
Eff: `(not (loaded` x_1`))`, `(free` x_2`)`
LOAD: $a_3(x_1, x_2, x_3)$

Static: $\neg(x_1{=}x_2)$, $\neg(x_1{=}x_3)$, $\neg(x_2{=}x_3)$
Pre: `(at` x_2`)`, `(at` x_3`)`, `(free` x_1`)`, `(free` x_3`)`, `(connected` x_2 x_3`)`
Eff: `(not (free` x_3`))`, `(loaded` x_2`)`
Objects: o_1, o_2, o_3, o_4 *(2 pallets × 2 locations)*

7.2 Forklift Pallet Management

Figure 7 reports the primary quantitative results on the real-world forklift evaluation. The neuro-symbolic framework achieves a strong success rate of 90% with 30 demonstrations, thus approaching the 98% ceiling set by the hand-engineered classical controller while requiring zero manual domain engineering. And it also maintains a high success rate of 86% with only 10 demonstration. The graceful degradation from 30 to 5 demonstrations (90% → 78% strong SR) reflects the diminishing density of training data in the ego-centric observation space.

Three findings deserve emphasis. **(i) The soft success rate remains consistently high.** The gap between strong and soft success (≈ 6–12 pp) is attributable primarily to minor fork-tip contacts during the alignment phase, not to planning failures or gross manipulation errors. This gap narrows as the number of demonstrations grows, consistent with the diffusion policy better covering the multi-modal distribution of approach trajectories. **(ii) Classical control provides an engineering ceiling, not a fair baseline.** The classical controller achieves 98% by relying on a precisely hand-crafted, fully specified alignment controller, a task-specific artefact that required months of expert engineering and encodes detailed kinematic and geometric knowledge of the forklift-pallet interaction. Our framework treats the control layer as a *prior*: low-level controllers handle command-tracking and physical stabilization, freeing the learned components to operate at a higher level of abstraction. The Oracle grounds each skill to the relevant object poses, determining *where* to move, while diffusion policies capture *how* to move by learning continuous motion behavior from demonstrations. At the top level, the symbolic planner determines *what* skill to execute and *when*, chaining operators into coherent task-level plans. The human role is thereby reduced from *engineering a complete task-level controller* to *demonstrating a skill*: rather than explicitly specifying preconditions, alignment logic, and control gains for each new task, the expert performs as few as one demonstration, and the framework distributes that demonstrated intent across the planning and policy layers automatically. A low-level motion controller remains a prerequisite, but its development or learning is a strictly one-time effort, once acquired, controls are shared across all tasks of the system. Importantly, this one-time cost is of a fundamentally different nature than task-level engineering: a motion controller need only capture the



a: 30 demonstrations b: 10 demonstrations c: 5 demonstrations

Fig. 9: Distribution of performance (success/failure) of the forklift pallet loading task over the forklift reset region of the state. Each episode reset pose is represented by an arrow parametrized with origin $(x_{\text{init}}, y_{\text{init}}, yaw_{\text{init}})$ relative to the pallet location. Blue arrows represent successes, while Red arrows represent failures. Reset configurations are sampled from a Gaussian distribution around a nominal pose. The three subfigures correspond to policies trained with 30, 10, and 5 demonstrations, respectively. No strong global correlation is observed between task success and initial pose. In the 5-demonstration setting, performance shows higher sensitivity to x, with a smooth success region of approximately 50 cm. Failures are primarily correlated with perception noise (lightning notably impacts the detections) and the number of demonstrations.

platform's physical capabilities, not the semantics of any particular task, object configuration, or alignment scenario. Every new skill, pallet type, or environment is then handled purely through demonstration, with no further control engineering required. The 8–12 pp gap between the two approaches thus quantifies the tradeoff between full expert specification and demonstration-driven learning: performance against generality and deployment speed. For reference, informal human trials suggest that operator performance on these tasks falls within the same range as our agent trained on 30 demonstrations. **(iii) The reset-pose distribution (Fig. 9) reveals that failures are driven by the number of training samples and perception rather than pose geometry.** No strong spatial correlation is observed between initial pose and task outcome within the demonstrated configuration space, suggesting robust generalization across the tested distribution. This diagnosis was made possible by the modular transparent design of our architecture: because perception, planning, and control are explicit, decoupled components, practitioners can isolate failure modes to a specific module and reason about their root cause directly, without resorting to black-box attribution methods. This interpretability has enormous practical value for deployment in safety-critical robotic systems and represents a distinct advantage over end-to-end learned policies, where such failure attribution is generally intractable.

7.3 Long-Horizon Composition and Rapid Adaptation

The long-horizon two-pallet, two-zone experiment demonstrates that compositional planning from learned operators incurs negligible overhead: the MetricFF planner produces the correct eight-operator sequence in under one second, and end-to-end task execution proceeds without any additional human intervention. Failures in this setting

are exclusively attributable to low-level execution errors (primarily perception and forks insertion within pallets collision), confirming that the symbolic layer correctly captures task structure even in compositions never seen during training.

The single-demonstration truck-bed adaptation experiment is particularly instructive. The fact that a single new demonstration, is sufficient to extend over an existing skill policy speaks to the modularity of the architecture. End-to-end imitation learning approaches would require collecting and labeling a new dataset for the entire task, retraining the full policy, and re-evaluating across all task configurations; our framework localizes the update to a single affected operator and reuses all previously acquired knowledge without modification.

7.4 Cross-Platform Arm Experiments

Results on the Kinova Gen3 benchmarks confirm the domain independence of the pipeline. The same framework, with no changes to symbolic abstraction, planning, or learning components (the perception only is different), produces valid, executable PDDL domains for both the stacking and kitchen environments. It is all the more important to note that our framework has been trained with *only a* single demonstration in both the kitchen and stacking tasks. It utilized this single demonstration to project and augment the data in the real world using existing controls, before abstracting automatically the planning domain (VLM annotation, graph construction, ASP solver) to solve the long-horizon problems. Crucially, unlike end-to-end approaches, our framework generalizes across novel task orderings and reuses the same controls across all tasks (inter-task generalization) without any additional demonstrations, whereas end-to-end methodologies require retraining for each new goal configuration.

Also, note the ASP solver's ability to represent non-spatial temporal operators (e.g., `wait`) Lorang et al. (2025a), yielding a structurally richer PDDL domain than clustering-based abstraction methods, which collapse all pre-conditions to purely spatial predicates.

8 Discussion

The results raise several points worth examining beyond the quantitative performance figures.

On the role of the classical controller. A recurring question in neuro-symbolic robotics is where the boundary between learned and engineered components should lie. Our results suggest that this boundary need not be fixed: by treating low-level controllers as fixed *priors* rather than as objects of learning, the framework gains reliable, sample-efficient low-level execution and concentrates the learning budget on higher-level structure, namely operator sequencing, precondition abstraction, and motion direction, where demonstrations are most informative. The residual gap to the classical controller's performance is therefore not a failure of the learning approach but a quantification of what hand-engineering uniquely supplies: tightly-tuned kinematic geometry and fine-grained alignment logic. Whether this gap is worth closing through further demonstration collection or through targeted engineering is a deployment decision rather than a methodological one.

On interpretability as an operational property. The modular decomposition into perception, planning, and control components produces interpretability as a practical side-effect rather than as a post-hoc explanation tool. In the forklift experiments, failure attribution required no gradient-based or surrogate-model analysis; failures could be traced directly to identifiable modules, specifically perception noise under varying illumination conditions or insufficient demonstration density in particular regions of the observation space. For safety-critical deployment, this diagnostic tractability is arguably as important as the success rate itself: an operator can act on a diagnosis (improve lighting, collect targeted demonstrations) in a way that is simply not possible when attribution relies on black-box methods.

On failure recovery. The current framework operates in open-loop at the task level: the symbolic plan is executed sequentially without replanning on failure. While the Transformer termination predictor absorbs low-level execution noise, mid-task perception failures or unexpected environmental changes can cause silent, unrecoverable downstream errors. Closing this loop requires a reliable symbolic state estimator, one capable of determining at each step whether the preconditions for the next operator are genuinely satisfied. This is technically non-trivial: VLM-based state estimation introduces its own failure modes, including ambiguous scenes and partial occlusion, and replanning under incorrect state estimates risks compounding errors. We regard this as the most pressing open problem for the framework's transition from demonstration-driven learning to genuinely reactive autonomy.

On data efficiency and its limits. The single-demonstration performance on the arm benchmarks and the graceful degradation curve on the forklift (from thirty to five demonstrations) both suggest that the framework's data efficiency stems from a combination of factors: data augmentation within existing control trajectories, the compressive effect of ego-centric observation filtering, and the ability of the symbolic planner to reuse acquired operators in novel compositions. However, data efficiency is not monotone in task complexity. Operators with multimodal approach distributions, where the same high-level action can be initiated from qualitatively different configurations, are underserved by small demonstration sets, as the diffusion policy cannot adequately cover the distribution's modes. This is consistent with the observed sensitivity to the number of training samples rather than to pose geometry, and it defines a natural criterion for deciding when additional demonstrations are warranted.

9 Limitations

Beyond failure recovery, discussed separately in Section 8, it is important to point out some additional limitations, notably "scaling up", a problem that bedevils all current task and skill learning approaches. While the proposed framework scales well for modest skill repertoires ($|\mathcal{O}| \leq 10$ in all experiments), but two bottlenecks emerge as the library grows. First, the ASP solver's search complexity increases with the number of graph edges, potentially making domain synthesis expensive or even intractable for large task libraries. A second problem is related to the VLM-based bisimulation compression which may conflate semantically distinct states whose visual

differences are subtle (e.g., two assembly states differing only in a single bolt orientation), leading to unsolvable satisfiability problems, i.e., no working ASP output, or an under-specified PDDL domain. Currently, human supervision over the graph construction is still necessary. Hierarchical domain construction, grouping operators into composable sub-domains, is a natural mitigation we leave to future work. Additionally, skill segmentation relies on an automatic change-point detector applied to the motion signal which may fail to segment skills with smooth, overlapping velocity profiles (e.g., a pick immediately followed by a place with no intermediate pause), negatively impacting sub-policy training. While the Transformer termination predictor partially absorbs this noise, fundamentally unreliable segmentation degrades both graph construction and diffusion policy training. A semantics-aware segmenter conditioned on VLM-predicted event boundaries would be a natural direction to improve robustness.

10 Conclusion

We have presented *Build on Priors* — leveraging foundation models, classical controls, and perception as structural priors — a scalable neuro-symbolic framework for data-efficient robot manipulation that learns both a symbolic planning domain and a set of neural skill policies from as few as one to thirty unannotated demonstrations per skill, without manual domain engineering or symbolic supervision. The framework integrates four tightly coupled components: VLM-driven automated annotation and bisimulation-based graph construction, ASP-based synthesis of expressive PDDL domains, oracle-guided ego-centric observation filtering, and control-level imitation learning with real-world data augmentation.

Empirical validation on a real autonomous industrial forklift and on a Kinova Gen3 robotic arm across stacking and kitchen benchmarks demonstrate that the proposed model produces interpretable, generalizable, and data-efficient behavior across qualitatively different robotic platforms and task domains. On the forklift, the neuro-symbolic framework achieves a strong success rate of 90% with 30 demonstrations and 86% with only 10, approaching the ceiling set by a fully hand-engineered classical controller while requiring only a human-provided skill vocabulary rather than full manual domain specification. Statistical evaluation over a Gaussian distribution of reset poses confirms that the framework maintains robust manipulation performance across a wide range of initial conditions, degrading gracefully toward the boundary of the feasibility region. Long-horizon experiments with two pallets and two drop-off zones demonstrate that compositional planning from learned operators generalizes to task structures never seen in training, and a single-demonstration truck-bed adaptation experiment shows that the modular symbolic architecture localizes new skill acquisition to the affected operator without disturbing previously learned knowledge.

Taken together, these results support the broader claim: *grounding control learning in symbolic reasoning, and grounding symbolic reasoning in perceptual abstraction driven by large pre-trained models, yields a practical path toward scalable, expert-free, and interpretable robot learning in real-world settings.* The proposed framework is neither a pure imitation learning method nor a pure symbolic planning system,

but rather a tightly integrated symbolic-subsymbolic hierarchy in which each layer strengthens the others. The symbolic planner decomposes the combinatorial task-planning problem into independently learnable sub-problems; the oracle reduces each sub-problem to its minimal ego-centric form; the diffusion policy solves each sub-problem from a compact, smooth learning target; and the VLM supplies the semantic supervision that makes automatic domain construction possible.

Looking forward, several directions appear particularly promising. First, closing the perception-to-planning loop via a reliable symbolic state estimator would enable genuine failure recovery and replanning, transforming the framework from an open-loop sequencer into a fully reactive task-and-motion planning system. Second, extending the oracle and data augmentation machinery to mobile manipulation platforms would broaden the range of deployable tasks significantly. Third, integrating the neuro-symbolic planning layer with large-scale VLA-style controllers for individual skills, using the symbolic planner for task-level sequencing and a VLA for rich visuomotor execution, represents a compelling hybrid that would combine the data efficiency and interpretability of our framework with the perceptual generality of foundation-model-based controllers.

A Appendix

A.1 VLM Graph Construction

The following figures illustrate the VLM-based graph construction pipeline. Fig. 10 shows the full state-transition graph produced by Gemini 3 Flash (DeepMind 2025) over 5 demonstrations per skill (*navigate*, *load*, *unload*). Each node represents a cluster of visually equivalent high-level states, determined by querying the VLM with pairs of observations and collecting a boolean `true`/`false` matching response. Edges are annotated with the single-word skill label associated with the transition in the demonstration trajectory. Three nodes were manually injected by a human operator to resolve ambiguities that the VLM could not confidently resolve, ensuring full graph connectivity. Fig. 11 provides a representative gallery of the raw visual observations assigned to each node, illustrating the degree of intra-node visual variability that the VLM is required to abstract over.

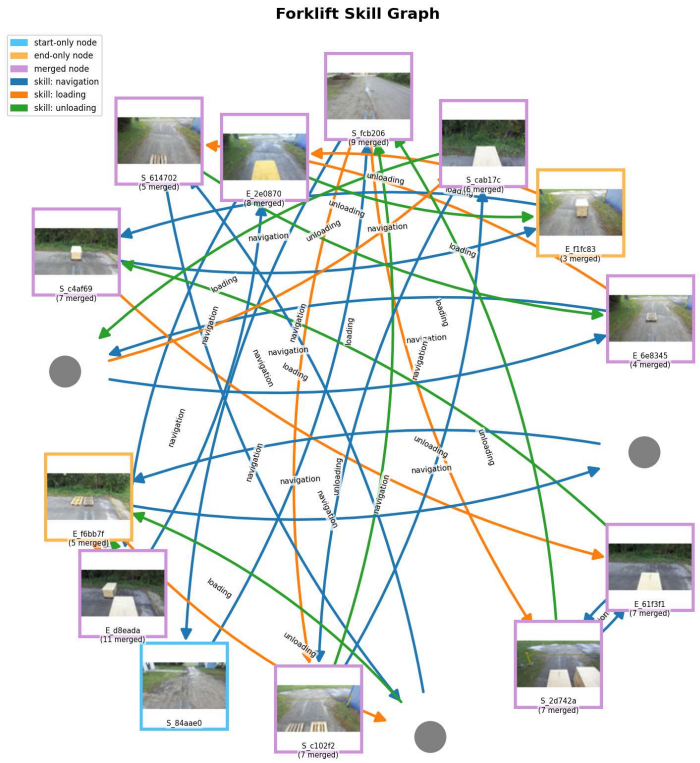


Fig. 10: State-transition graph constructed by Gemini 3 Flash over 5 demonstrations per skill. Nodes represent clusters of visually equivalent high-level states (boolean VLM matching); edges are annotated with single-word skill labels. Three nodes were manually completed to ensure full connectivity.

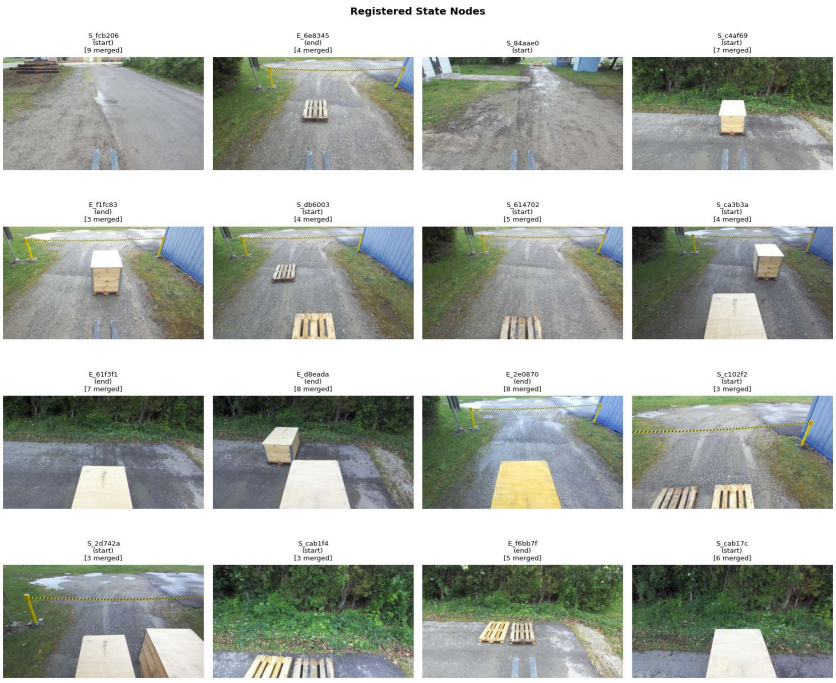


Fig. 11: Gallery of raw visual observations grouped by node. Each cell shows the set of demonstration frames the VLM merged into a single abstract state, illustrating the degree of intra-node visual variability handled by the matching mechanism.

A.2 Gross ASP Outputs

Forklift Multiple Pallets Storage Domain

The following action models are the raw symbolic outputs of the ASP solver applied to the bisimulation-reduced state-transition graphs described above. Three graph configurations are reported: the full graph, a graph with one node and two edges missing, and a graph with two nodes and six edges missing. Small graph incompletion can induce action models becoming less compact or noisier, requiring more action parameters, additional preconditions, and finer-grained predicate structure, yet remain executable as planning domains.

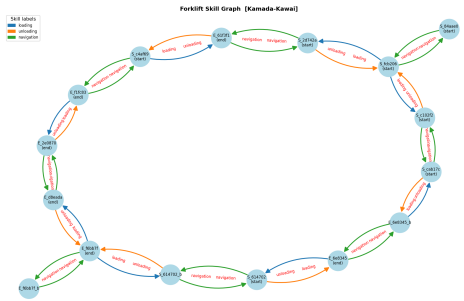


Fig. 12: Bisimulation reduction of the full state-transition graph fed to the ASP solver. Behaviourally equivalent nodes are merged into canonical representatives prior to induction.

Full Domain Graph

MOVE: $a_1(x_1, x_2)$ Static: $\neg(x_1{=}x_2)$
Pre: $p_2(x_1),\ p_3(x_2)$
Eff: $-p_2(x_1),\ -p_3(x_2),\ +p_3(x_1),\ +p_2(x_2)$

UNLOAD: $a_2(x_1, x_2)$
Pre: $p_3(x_2),\ p_4(x_1)$
Eff: $-p_4(x_1),\ +p_1(),\ +p_5(x_1, x_2)$

LOAD: $a_3(x_1, x_2)$
Pre: $p_1(),\ p_3(x_2),\ p_5(x_1, x_2)$
Eff: $-p_1(),\ -p_5(x_1, x_2),\ +p_4(x_1)$

Objects: o_1, o_2, o_3, o_4 *(2 pallets × 2 locations)*

One Node, Two Edges Missing

MOVE: $a_1(x_1, x_2, x_3)$

Static: $\neg(x_1{=}x_2),\ \neg(x_1{=}x_3)$
Pre: $p_3(x_2, x_1),\ p_3(x_2, x_2),\ p_4(x_3, x_1)$
Eff: $-p_3(x_2, x_1),\ -p_3(x_2, x_2),\ +p_3(x_1, x_1),\ +p_3(x_1, x_2)$

UNLOAD: $a_2(x_1, x_2)$

Static: $\neg(x_1{=}x_2)$
Pre: $p_2(x_1),\ p_3(x_1, x_1),\ p_3(x_2, x_2),\ p_4(x_2, x_2)$
Eff: $-p_2(x_1),\ -p_3(x_2, x_2),\ +p_1(),\ +p_3(x_1, x_2)$

LOAD: $a_3(x_1, x_2)$

Static: $\neg(x_1{=}x_2)$
Pre: $p_1(),\ p_3(x_2, x_1),\ p_3(x_2, x_2),\ p_4(x_1, x_1)$
Eff: $-p_1(),\ -p_3(x_2, x_1),\ +p_2(x_2),\ +p_3(x_1, x_1)$

Objects: o_1, o_2, o_3, o_4 *(2 pallets × 2 locations)*

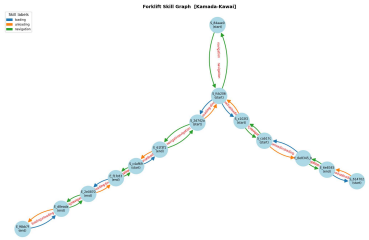


Fig. 13: Bisimulation reduction of the partial graph (two nodes, six edges missing). The reduced node set causes the solver to introduce additional parameters and finer-grained predicates.

Two Nodes, Six Edges Missing

MOVE: $a_1(x_1, x_2, x_3)$

Static: $\neg(x_1{=}x_2)$
Pre: $p_1(x_1),\ p_2(x_3),$
$p_4(x_1, x_1),\ p_4(x_2, x_2),\ p_4(x_3, x_1)$
Eff: $-p_1(x_1),\ +p_1(x_2)$

UNLOAD: $a_2(x_1, x_2)$

Static: $\neg(x_1{=}x_2)$
Pre: $p_1(x_1),\ p_1(x_2),\ p_3(x_1),\ p_4(x_1, x_2)$
Eff: $-p_3(x_1),\ +p_2(x_2)$

LOAD: $a_3(x_1, x_2, x_3)$

Static: $\neg(x_1{=}x_2),\ \neg(x_1{=}x_3),\ \neg(x_2{=}x_3)$
Pre: $p_1(x_2),\ p_1(x_3),\ p_2(x_1),\ p_2(x_3),\ p_4(x_2, x_3)$
Eff: $-p_2(x_3),\ +p_3(x_2)$

Objects: o_1, o_2, o_3, o_4 *(2 pallets × 2 locations)*

References

Ahmetoglu A, Seker MY, Piater J, et al (2022) Deepsym: Deep symbol generation and rule learning from unsupervised continuous robot interaction for planning. Journal of Artificial Intelligence Research 75:709–745. https://doi.org/10.1613/jair.1.13754, arXiv:2012.02532 [cs]

Beleznai C, Reisinger L, Pointner W, et al (2025) Pallet Detection and 3D Pose Estimation via Geometric Cues Learned from Synthetic Data. In: Proceedings of the Conference of the Hungarian Association for Image Analysis and Pattern Recognition

Black K, Brown N, Driess D, et al (2024) π_0: A vision-language-action flow model for general robot control. arXiv preprint arXiv:241024164 URL https://arxiv.org/abs/2410.24164

Bommasani R, Hudson DA, Adeli E, et al (2021) On the opportunities and risks of foundation models. arXiv preprint arXiv:210807258 URL https://arxiv.org/abs/2108.07258

Bonet B, Geffner H (2020) Learning First-Order Symbolic Representations for Planning from the Structure of the State Space, IOS Press, Santiago de Compostela, Spain, pp 2322–2329. Frontiers in Artificial Intelligence and Applications, https://doi.org/10.3233/FAIA200361, URL https://doi.org/10.3233/FAIA200361

Brohan A, Brown N, Carbajal J, et al (2023) RT-2: Vision-language-action models transfer web knowledge to robotic control. In: Conference on Robot Learning (CoRL), URL https://arxiv.org/abs/2307.15818

Cheng S, Xu D (2023) League: Guided skill learning and abstraction for long-horizon manipulation. IEEE Robotics and Automation Letters 8(10):6451–6458. https://doi.org/10.1109/LRA.2023.3308061

Chi C, Feng S, Du Y, et al (2023) Diffusion policy: Visuomotor policy learning via action diffusion. In: Proceedings of Robotics: Science and Systems (RSS)

Chitnis R, Silver T, Tenenbaum JB, et al (2022) Learning neuro-symbolic relational transition models for bilevel planning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

Curtis A, Silver T, Tenenbaum JB, et al (2022) Discovering state and action abstractions for generalized task and motion planning. Proceedings of the AAAI Conference on Artificial Intelligence 36(5):5377–5384. https://doi.org/10.1609/aaai.v36i5.20475

DeepMind G (2025) Gemini 3 flash: Frontier intelligence built for speed. https://blog.google/products/gemini/gemini-3-flash/, accessed: 2026-03-28

Garrett CR, Paxton C, Lozano-Perez T, et al (2020) Online replanning in belief space for partially observable task and motion problems. In: IEEE International Conference on Robotics and Automation (ICRA), pp 5678–5684, URL https://arxiv.org/abs/1911.04577

Garrett CR, Chitnis R, Holladay R, et al (2021) Integrated task and motion planning. Annual Review of Control, Robotics, and Autonomous Systems 4(Volume 4, 2021):265–293. https://doi.org/10.1146/annurev-control-091420-084139

Gehring C, Asai M, Chitnis R, et al (2022) Reinforcement learning for classical planning: Viewing heuristics as dense reward generators. Proceedings of the International Conference on Automated Planning and Scheduling 32(1):588–596. https://doi.org/10.1609/icaps.v32i1.19846

Goel S, Shukla Y, Sarathy V, et al (2022) Rapid-learn: A framework for learning to recover for handling novelties in open-world environments. In: IEEE ICDL

Guan L, Sreedharan S, Kambhampati S (2022) Leveraging approximate symbolic models for reinforcement learning via skill diversity

Hoffmann J (2003) The metric-ff planning system: Translating"ignoring delete lists"to numeric state variables. Journal of artificial intelligence research 20:291–341

Huang B, Yue L, Li B, et al (2024) VLM See, Robot Do: Human demo video to robot action plan via vision language model. arXiv preprint arXiv:241008792 URL https://arxiv.org/abs/2410.08792

Huemer J, Murschitz M, Schörghuber M, et al (2025) ADAPT: An Autonomous Forklift for Construction Site Operation. URL https://arxiv.org/abs/2503.14331

Hussein A, Gaber MM, Elyan E, et al (2017) Imitation learning: A survey of learning methods. ACM Comput Surv 50(2):21:1–21:35. https://doi.org/10.1145/3054912

Icarte RT, Klassen TQ, Valenzano RA, et al (2020) Reward machines: Exploiting reward function structure in reinforcement learning. JAIR 73:173–208

Illanes L, Yan X, Icarte RT, et al (2020) Symbolic plans as high-level instructions for reinforcement learning. Proceedings of the International Conference on Automated Planning and Scheduling 30:540–550. https://doi.org/10.1609/icaps.v30i1.6750

Jocher G, Chaurasia A, Qiu J (2023) Ultralytics yolov8. URL https://github.com/ultralytics/ultralytics

Kaelbling LP, Lozano-Pérez T (2011) Hierarchical task and motion planning in the now. In: 2011 IEEE International Conference on Robotics and Automation, p 1470–1477, https://doi.org/10.1109/ICRA.2011.5980391, URL https://ieeexplore.ieee.org/abstract/document/5980391/

Keller L, Tanneberg D, Peters J (2025) Neuro-symbolic imitation learning: Discovering symbolic abstractions for skill learning. IEEE International Conference on Robotics and Automation (ICRA), arXiv:2503.21406, https://doi.org/10.48550/arXiv.2503.21406, URL http://arxiv.org/abs/2503.21406, arXiv:2503.21406 [cs]

Kim MJ, Pertsch K, Karamcheti S, et al (2024) OpenVLA: An open-source vision-language-action model. In: Conference on Robot Learning (CoRL), URL https://arxiv.org/abs/2406.09246

Kokel H, Manoharan A, Natarajan S, et al (2021) Reprel: Integrating relational planning and reinforcement learning for effective abstraction. In: ICAPS

Konidaris G, Kaelbling LP, Lozano-Perez T (2018) From skills to symbols: Learning symbolic representations for abstract high-level planning. Journal of Artificial Intelligence Research 61:215–289. https://doi.org/10.1613/jair.5575

Kumar N, McClinton W, Chitnis R, et al (2023) Learning efficient abstract planning models that choose what to predict. arXiv, arXiv:2208.07737, URL http://arxiv.org/abs/2208.07737, arXiv:2208.07737 [cs]

Le H, Jiang N, Agarwal A, et al (2018) Hierarchical imitation and reinforcement learning. In: Proceedings of the 35th International Conference on Machine Learning. PMLR, Stockholm, Sweden, pp 2917–2926, URL https://proceedings.mlr.press/v80/le18a.html

Lorang P, Goel S, Shukla Y, et al (2024a) A framework for neurosymbolic goal-conditioned continual learning in open world environments. In: 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), p 12070–12077, https://doi.org/10.1109/IROS58592.2024.10801627, URL https://ieeexplore.ieee.org/document/10801627

Lorang P, Horvath H, Kietreiber T, et al (2024b) Adapting to the "open world": The utility of hybrid hierarchical reinforcement learning and symbolic planning. In: 2024 IEEE International Conference on Robotics and Automation (ICRA), p 508–514, https://doi.org/10.1109/ICRA57147.2024.10611594, URL https://ieeexplore.ieee.org/document/10611594

Lorang P, Lu H, Huemer J, et al (2025a) Few-shot neuro-symbolic imitation learning for long-horizon planning and acting. In: Proceedings of The 9th Conference on Robot Learning. PMLR, Seoul, p 2501–2518, URL https://proceedings.mlr.press/v305/lorang25a.html

Lorang P, Lu H, Scheutz M (2025b) Curiosity-driven imagination: Discovering plan operators and learning associated policies for open-world adaptation. IEEE International Conference on Robotics and Automation (ICRA), Atlanta, USA, arXiv:2503.04931, https://doi.org/10.48550/arXiv.2503.04931, URL http://arxiv.org/abs/2503.04931, arXiv:2503.04931 [cs]

Loula J, Allen K, Silver T, et al (2020) Learning constraint-based planning models from demonstrations. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, Las Vegas, NV, USA, pp 5410–5416, URL https://ieeexplore.ieee.org/abstract/document/9341535/

Mahalanobis PC (2018) Reprint of: "On the Generalised Distance in Statistics". Sankhyā A 80:1–7. https://doi.org/10.1007/s13171-019-00164-5

Mandlekar A, Nasiriany S, Wen B, et al (2023) MimicGen: A data generation system for scalable robot learning using human demonstrations. In: Conference on Robot Learning (CoRL), URL https://arxiv.org/abs/2310.17596

Manschitz S, Kober J, Gienger M, et al (2014) Learning to sequence movement primitives from demonstrations. 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems p 4414–4421. https://doi.org/10.1109/IROS.2014.6943187

McDermott D, Ghallab M, Howe A, et al (1998) PDDL - The Planning Domain Definition Language

Minderer M, Gritsenko A, Houlsby N (2024) Scaling open-vocabulary object detection. https://doi.org/10.48550/arXiv.2306.09683, URL http://arxiv.org/abs/2306.09683, arXiv:2306.09683 [cs]

Mitchener L, Tuckey D, Crosby M, et al (2022) Detect, understand, act: A neuro-symbolic hierarchical reinforcement learning framework. Machine Learning 111(4):1523–1549. https://doi.org/10.1007/s10994-022-06142-7

Newell A, Simon HA (1976) Computer science as empirical inquiry: Symbols and search. Communications of the ACM 19(3):113–126. https://doi.org/10.1145/360018.360022

Open X-Embodiment Collaboration (2023) Open X-Embodiment: Robotic learning datasets and RT-X models. URL https://arxiv.org/abs/2310.08864

Osa T, Pajarinen J, Neumann G, et al (2018) An algorithmic perspective on imitation learning. Foundations and Trends in Robotics 7:1–179. https://doi.org/10.1561/2300000053

Pertsch K, Lee Y, Wu Y, et al (2021) Demonstration-guided reinforcement learning with learned skills

Rajaraman N, Yang LF, Jiao J, et al (2020) Toward the fundamental limits of imitation learning. arXiv, arXiv:2009.05990, https://doi.org/10.48550/arXiv.2009.05990, URL http://arxiv.org/abs/2009.05990, arXiv:2009.05990 [cs]

Rodriguez ID, Bonet B, Romero J, et al (2021) Learning First-Order Representations for Planning from Black Box States: New Results. In: Proceedings of the 18th

International Conference on Principles of Knowledge Representation and Reasoning, pp 539–548, https://doi.org/10.24963/kr.2021/51, URL https://doi.org/10.24963/kr.2021/51

Sarathy V, Kasenberg D, Goel S, et al (2021) Spotter: Extending symbolic planning operators through targeted reinforcement learning. In: AAMAS

Shah N, Nagpal J, Verma P, et al (2024) From reals to logic and back: Inventing symbolic vocabularies, actions, and models for planning from raw data. arXiv, arXiv:2402.11871, https://doi.org/10.48550/arXiv.2402.11871, URL http://arxiv.org/abs/2402.11871, arXiv:2402.11871 [cs]

Shridhar M, Manuelli L, Fox D (2022) Perceiver-actor: A multi-task transformer for robotic manipulation. In: Conference on Robot Learning (CoRL), URL https://arxiv.org/abs/2211.11736

Siciliano B, Sciavicco L, Villani L, et al (2009) Robotics: Modelling, Planning and Control. Springer, London, URL https://link.springer.com/book/10.1007/978-1-84628-642-1

Silver T, Chitnis R, Kumar N, et al (2022) Predicate invention for bilevel planning. arXiv, arXiv:2203.09634, URL http://arxiv.org/abs/2203.09634, arXiv:2203.09634 [cs]

Silver T, Athalye A, Tenenbaum JB, et al (2023) Learning neuro-symbolic skills for bilevel planning. In: Proceedings of The 6th Conference on Robot Learning. PMLR, Auckland, New Zealand, p 701–714, URL https://proceedings.mlr.press/v205/silver23a.html

Sutton RS, Precup D, Singh S (1999) Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence https://doi.org/https://doi.org/10.1016/S0004-3702(99)00052-1

Tanwani AK, Yan A, Lee J, et al (2021) Sequential robot imitation learning from observations. The International Journal of Robotics Research 40(10–11):1306–1325. https://doi.org/10.1177/02783649211032721

Teng S, Chen L, Ai Y, et al (2023) Hierarchical interpretable imitation learning for end-to-end autonomous driving. IEEE Transactions on Intelligent Vehicles 8(1):673–683. https://doi.org/10.1109/TIV.2022.3225340

Umili E, Antonioni E, Riccio F, et al (2021) Learning a symbolic planning domain through the interaction with continuous environments

Vaswani A, Shazeer N, Parmar N, et al (2017) Attention is all you need. In: Advances in Neural Information Processing Systems (NeurIPS), URL https://arxiv.org/abs/1706.03762

Wake N, Kanehira A, Sasabuchi K, et al (2024) GPT-4V(ision) for robotics: Multimodal task planning from human demonstration. In: IEEE Robotics and Automation Letters, URL https://arxiv.org/abs/2311.12015

Wolfe J, Marthi B, Russell S (2010) Combined task and motion planning for mobile manipulation. Proceedings of the International Conference on Automated Planning and Scheduling 20:254–257. https://doi.org/10.1609/icaps.v20i1.13436

Xu T, Li Z, Yu Y (2020) Error bounds of imitating policies and environments. arXiv, arXiv:2010.11876, https://doi.org/10.48550/arXiv.2010.11876, URL http://arxiv.org/abs/2010.11876, arXiv:2010.11876 [cs]

Yang F, Lyu D, Liu B, et al (2018) Peorl: Integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. pp 4860–4866, https://doi.org/10.24963/ijcai.2018/675

Yuan Y, Wang H, Zhang R (2023) Look before you leap: Unveiling the power of GPT-4V in robotic vision-language planning. arXiv preprint arXiv:231117842 URL https://arxiv.org/abs/2311.17842

Zare M, Kebria PM, Khosravi A, et al (2024) A survey of imitation learning: Algorithms, recent developments, and challenges. IEEE Transactions on Cybernetics 54(12):7173–7186. https://doi.org/10.1109/TCYB.2024.3395626

Zhao TZ, Kumar V, Levine S, et al (2023) Learning fine-grained bimanual manipulation with low-cost hardware. In: Proceedings of Robotics: Science and Systems (RSS)

Zhu Y, Stone P, Zhu Y (2022) Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. IEEE Robotics and Automation Letters 7(2):4126–4133. https://doi.org/10.1109/LRA.2022.3146589

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- NeuroSymbolic1DemoUnloadTruckspeedx4.mp4
- NeuroSymbolic30DemosStatisticalAnalysisForklift.m4v