In ICAPS 2026 Workshop on Hierarchical Planning (HPlan 2026)
Dublin, Ireland

Learning HTNs from Visual Demonstration with Vision-Language Models: Preliminary Results

Ngoc La[1], **Karthik Mahadevan**[1], **Pulkit Verma**[2], **Julie A. Shah**[1]
[1]Massachusetts Institute of Technology
[2]Indian Institute of Technology Madras
{ntmla, mahade0}@mit.edu, pulkitv@cse.iitm.ac.in, julie_a_shah@csail.mit.edu

Abstract

Hierarchical task networks (HTNs) are a popular formalism in automated planning, enabling complex, long-horizon tasks to be solved through hierarchical decomposition. However, constructing HTN models manually is difficult and requires domain expertise. While vision-language models (VLMs) have been used to generate planning representations in Planning Domain Definition Language (PDDL), their application to learning HTNs from raw visual demonstrations and generating domains in Hierarchical Domain Definition Language (HDDL) remains unexplored. In this work, we present Vision2HTN, a pipeline that uses VLMs to learn HTN methods directly from visual inputs together with existing PDDL domains, augmented with iterative HDDL syntax validation and repair. We evaluate our approach on Blocksworld and Overcooked, finding that the pipeline can generate usable HDDL domains from visual demonstrations and given PDDL domains. Performance is substantially stronger in Blocksworld than Overcooked, pre-defined task information is especially helpful in complex domains, and the benefit of visual input depends on domain complexity and demonstration type. These results suggest VLMs are a promising direction for HTN learning from visual data, while highlighting challenges in robustness and domain complexity.

1 Introduction

Hierarchical task networks (HTNs) are a promising formalism for planning in complex, long-horizon tasks (Erol, Hendler, and Nau 1995). By organizing knowledge into hierarchical decompositions of tasks and methods, HTN planners can exploit domain structure to narrow the search space while remaining interpretable to human users. They also provide a general way to encode problem-solving strategies in the domain model, allowing procedural knowledge to be reused across different problem instances. Despite these benefits, HTN planners rely on the assumption that this hierarchical structure is already well-defined, yet constructing such a structure is often difficult and requires expertise in both the application domain and the planning language. Since HTN structures are themselves human-friendly, especially with the formally defined Hierarchical Domain Definition Language (HDDL) (Höller et al. 2020), automatically learning HTN models from data is an essential direction for bringing HTN planning to more real-world applications.

Prior approaches to learning HTN models have mainly relied on statistical and analytical strategies to interpret plan traces into HTN methods (Hogg, Munoz-Avila, and Kuter 2008; Chen et al. 2021; Li et al. 2024; Langley 2022). While these methods can offer correctness and completeness guarantees for the generated HTN models, they require a substantial amount of plan trace data in symbolic form, and some also require additional manual annotation effort. Recently, large language models (LLMs) have opened a generative path to HTN learning, where models are synthesized from natural language descriptions or symbolic plan traces rather than derived analytically (Fine-Morris et al. 2025; Muñoz Avila, Aha, and Rizzo 2025; Xu and Munoz-Avila 2025). However, the inputs to these approaches remain predominantly symbolic, in the form of plan traces from classical planners represented in the Planning Domain Definition Language (PDDL) (McDermott et al. 1998).

To our knowledge, learning HTN models from raw visual demonstration data remains underexplored, even though such data is abundant, naturally collected from human performers, and widely available online. The closest related work is by Xi, Gould, and Thiébaux (2024), who learn lifted action models from visual traces using neuro-symbolic methods; however, they do not address hierarchical structure, which we build upon here. The advent of vision-language models (VLMs) offers a promising path forward. VLMs have demonstrated strong capabilities in translating visual observations into planning-relevant representations, including generating PDDL problem files, grounding or creating predicates from visual states, and synthesizing domain structure from scratch (Shirai et al. 2024; Hao et al. 2026; Zhang et al. 2025). Yet, to our knowledge, their application to HDDL generation and HTN learning remains unexplored.

In this work, we investigate the use of VLMs for learning HTNs directly from visual demonstration data together with given PDDL domains. We make the following contributions:

- Vision2HTN, a novel pipeline with syntax validation, in which a VLM generates HTNs in HDDL format from visual demonstrations and PDDL, augmented with iterative repair via existing HDDL parser tools.
- An empirical analysis across three factors: presence of pre-defined task specifications, the role of visual inputs, and VLM choice, assessing their effect on HTN generation quality using 2D simulation data in two domains.

2 Related Work

Learning HTN Models from Plan Traces. The problem of automatically constructing HTN domain models from observed behavior has a rich history. HTN-Maker (Hogg, Munoz-Avila, and Kuter 2008) introduced one of the earliest systematic approaches, learning HTN methods with minimal additional knowledge engineering by analyzing sequences of ground actions and identifying recursive task decompositions from labeled plan traces. Zhuo, Muñoz Avila, and Yang (2014) extended this line of work to the partially observed setting, learning HTN domains from plan traces where intermediate traces might be missing, using a MAX-SAT formulation to recover a consistent set of methods. Li et al. (2024) use a landmark-based approach to automatically identify subtask boundaries and derive HTN methods without requiring manual annotation of task hierarchies. Chen et al. (2021) address the setting where demonstrations are available but not labeled with task annotations, learning hierarchical task networks with preferences from unannotated robot demonstrations by clustering observed behavior and inferring decomposition structure. Langley (2022) approaches the problem from an inductive logic programming perspective, learning hierarchical problem networks that capture recursive problem-solving structure from positive examples. More recently, Langley (2025) characterizes the HTN learning problem in terms of three subproblems, (i) identifying hierarchical structure, (ii) unifying method heads, and (iii) finding method conditions, and poses open challenges for future work. A common limitation across these approaches is their dependence on symbolic plan trace data provided in a planning language, and several additionally require either manual task annotation or a pre-specified primitive action vocabulary. Our work differs fundamentally in that the input is raw visual data with no assumption of access to symbolic traces. The closest work to ours in this regard is Xi, Gould, and Thiébaux (2024), who use neuro-symbolic methods to learn lifted action models from visual traces. However, their approach does not produce hierarchical task structure, which is the primary focus of Vision2HTN, and could serve as a natural baseline for the action-level component of our pipeline.

LLM-Based HTN Generation. Large language models have more recently been applied to HTN model generation in a generative rather than analytical fashion. Fine-Morris et al. (2025) propose leveraging LLMs to generate hierarchical planning models informed by task-relevant natural language documents, demonstrating that document-grounded prompting can produce plausible HDDL structure. ChatHTN (Muñoz Avila, Aha, and Rizzo 2025) interleaves approximate LLM-based decomposition with symbolic HTN planning, using the LLM to propose method candidates that are then validated and refined by a classical HTN planner. Xu and Munoz-Avila (2025) develop an online learning approach in which HTN methods are incrementally refined during execution, integrating LLM-generated suggestions with symbolic plan verification. While these approaches demonstrate the viability of generative methods for HTN learning, they all rely on symbolic or textual inputs and do not address the case where task knowledge must be extracted from raw perceptual data.

Learning PDDL from Visual Input. Recent work has explored automating the construction of symbolic planning representations, such as PDDL, from visual input.

Prior work has explored the feasibility of leveraging visual inputs for generating PDDL problem files. ViLaIn can generate problem files from a scene observation and language instruction using a VLM and LLM in sequence (Shirai et al. 2024). Image2PDDL simplifies this into three sequential VLM calls: translating the initial state image, translating the goal state, and generating the problem file from both alongside the domain and few-shot examples (Dang, Kudláčková, and Edelkamp 2025). However, problem generation can be fragile, and ViLaIn-TAMP addresses this by introducing a verification and repair step (Siburian et al. 2025).

VLMs have been used to ground existing predicates by evaluating them through visual question answering tasks (Zhang et al. 2025; Merler et al. 2025). Recent work has also explored inventing new predicates from visual data through VLMs. VisualPredicator invents neuro-symbolic predicates implemented as VLM-queryable functions through online environment interaction (Liang et al. 2025). SkillWrapper establishes a formal theory of generative predicate invention that provides provable soundness and completeness guarantees for symbolic planning via active, coverage-based data collection (Yang et al. 2025). pix2pred proposes a large candidate pool of visual predicates from a small number of human demonstrations and then optimizes them to ensure downstream planning efficiency (Athalye et al. 2026).

Most prior work assumes the existence of PDDL domain files, which typically requires human expertise or extensive environmental interaction. Recent work has explored generating domain files from visual input. ROSAME-I (Xi, Gould, and Thiébaux 2024) learns flat PDDL action models from visual traces via a differentiable neuro-symbolic formulation, but requires per-domain training and per-step action labels. VLMFP uses a dual-VLM approach to autonomously generate the PDDL domain as well as the problem file by iteratively refining symbolic rules against simulating actions (Hao et al. 2026). UniDomain learns a unified PDDL domain from robot manipulation video keyframes using closed-loop refinement (Ye et al. 2025). Despite progress on generating flat PDDL representations from visual input, constructing HTN models from visual demonstrations remains unexplored. We extend this line of work by studying how HTN structure can be learned directly from visual demonstrations and added on top of an existing PDDL domain to produce an HDDL model.

3 HDDL

HDDL (Höller et al. 2020) is an extension of PDDL (McDermott et al. 1998) to model hierarchical strategies in the domains, and they define the domain and problem as:

Definition of Planning Domain: A planning domain $\mathcal{D}_{HDDL}$ is a tuple (L, T_P, T_C, M) defined as follows.

- L is the underlying predicate logic (similar to PDDL).
- T_P and T_C are finite sets of primitive and compound tasks, respectively.
- M is a finite set of decomposition methods with compound tasks from T_C and task networks over the set $T_P \cup T_C$.

Definition of Planning Problem: A planning problem is a tuple $\mathcal{P}_{HDDL} = (\mathcal{D}_{HDDL}, s_I, tn_I, g)$, where:

- s_I is the initial state, a ground conjunction of positive literals over the predicates assuming the closed world assumption (similar to PDDL).
- tn_I is the initial task network that may not necessarily be grounded.
- g is the goal description, being a first-order formula over the predicates (not necessarily grounded) (similar to PDDL).

A solution to $\mathcal{P}_{HDDL}$ is a sequence of primitive actions $\pi = (a_1, a_2, \ldots, a_n)$ drawn from T_P that can be produced by decomposing tn_I via methods in M, such that executing π from s_I produces a state satisfying g.

In other words, beyond the equivalent action definition, which establishes the rules of interaction with the environment, HDDL introduces two additional operators: *task* in T_P and *method* in M. In HDDL, a *task* represents a high-level action, while a *method* is a strategy to accomplish a task. Multiple methods can exist to perform a single task. Essentially, a method is a task network that decomposes a high-level task into a partially or totally ordered list of tasks and actions, which is also known as a hierarchical task network (HTN). Additionally, in HDDL problems, state-based goal definition is optional. Most HDDL solvers focus on HTN goals, which are defined as a list of goal tasks tn_I in the HDDL problem file.

In our work, we study how to generate HTNs for HDDL domains from visual demonstrations and given PDDL domains using VLMs. Our main focus is on automatically generating the HDDL domain, while the corresponding HDDL problems are still constructed manually for evaluation. Although problem construction is not the primary focus of this work, the quality of the generated HTNs also influences how naturally and effectively goals can be represented in HDDL problem files. As a result, improving HTN generation is important not only for domain modeling itself, but also for enabling more reliable evaluation of the generated domains.

4 Vision2HTN

Vision2HTN directly uses VLMs to generate HTNs in HDDL format and then refines the outputs through syntax validation with HDDL-Parser (Yousefi and Bercher 2025) and the PANDA parser (Höller et al. 2021).

Inputs: visual demonstrations, PDDL domain, and optional pre-defined tasks T_C as additional context.

Output: a syntactically correct HDDL domain $\mathcal{D}_{HDDL}$.

4.1 Structure

As illustrated in Figure 1, Vision2HTN takes demonstration videos or images together with a PDDL domain as input.

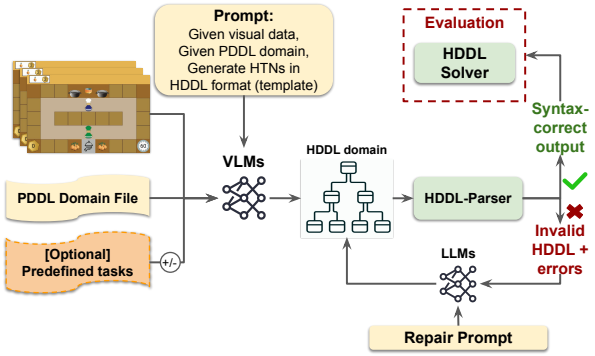

Figure 1: Structure of Vision2HTN Framework.

These inputs are paired with a carefully designed prompt, shared across all domains and described in Section 4.2, that asks the VLM to generate tasks and methods in HDDL format. In addition to these default inputs, the pipeline also supports optional context data; in this work, this context is a set of pre-defined tasks T_C used to help guide the VLMs in generating the HTNs. In this work, the given PDDL domains and the sets of pre-defined tasks are drawn from HTN-Maker (Hogg, Munoz-Avila, and Kuter 2008) or are manually crafted by us, and the generated output is a new HDDL domain file that includes the basic PDDL elements together with the generated HTNs.

Next, the generated HDDL domain is sent to a syntax-validation loop, where it is checked by the HDDL parsers mentioned above for syntax correctness. When syntax errors occur, both the error messages and the invalid HDDL domain are fed into a large language model (LLM) for iterative correction using the repair prompt provided in Section 4.2. The choice of LLM for this repair step can also be treated as a variant. While using the same model as the original VLM is a natural choice, using a different language model may help avoid repeating errors caused by model-specific biases. In our preliminary experiments, however, the repair step was mainly limited to fixing minor syntax errors and typos, and did not substantially improve the core structure of the generated tasks and methods. Therefore, to simplify the process and keep it uniform across settings, we used Claude Sonnet 4.6 for the repair step in all configurations. In future work, when repair also involves improving HTN structure to better ensure completeness and semantic correctness, the repair model can be varied between different VLMs in each setting.

Once the generated HDDL domain becomes syntactically valid, it is ready to go through the evaluation process described in Section 5.3. Overall, the Vision2HTN framework provides a first step toward studying HTN generation from visual demonstrations. While this initial framework is simple and intentionally avoids heavy additional engineering, we keep it minimal to better assess the capability of VLMs to generate hierarchical strategies from visual demonstrations. As this work evolves, we plan to further develop the framework based on our preliminary results to improve its performance.

4.2 Prompts

The actual prompts for generating HTNs and repairing any syntax errors are domain-independent and provided in the public GitHub repository: `github.com/Vision2HTN/vision2htn-assets`. Due to space limitations, we provide a summary of the prompt below that captures its main components. In the actual prompt, some core requirements are intentionally repeated to better ensure that the VLM follows them.

Domain Induction Prompt

The provided visual data show example demonstrations of the {`domain name`} domain. Treat these as illustrative instances only. Your goal is to infer general *lifted* HTN structures that apply broadly to the domain, not just to the specific objects, configurations, or goals shown. The generated output must satisfy the following requirements:

1. Include root tasks that capture general objectives in the domain.
2. Ensure generality: the generated domain should remain valid for different problems, not only the demonstrated ones.
3. Include coordination in multi-agent settings when appropriate.
4. Add new lifted predicates only when needed.
5. Ensure that all generated constructs are in valid HDDL format.
6. Return only the HDDL domain, with inline comments explaining the reasons for generation, and no markdown or code fences outside the HDDL content.

Here is the repair prompt that is used to ask the LLM or VLM to fix syntax errors in the invalid HDDL domains:

Syntax Error Correction Prompt

Here is the HDDL domain: {`HDDL domain`}
This HDDL domain file contains syntax errors: {`error messages`}
Please fix the syntax errors while preserving the intended hierarchical structure and semantics of the tasks and methods. Return only the corrected HDDL content without explanations or comments.

5 Experiments

5.1 Domains

We include the basic domain data, such as PDDL files, pre-defined tasks, visual data (videos and extracted frames), and related materials, in a public GitHub repository: `github.com/Vision2HTN/vision2htn-assets`. Since the experimental study is still ongoing and we plan to further develop this work into a full conference or journal paper, we leave the full codebase for a later release.

Blocksworld is a well-known benchmark domain in PDDL and HDDL, where an agent rearranges blocks on a table to reach a target configuration. The PDDL domain of Blocksworld and the set of pre-defined compound tasks are from the HTN-Maker work (Hogg, Munoz-Avila, and Kuter 2008). List of actions of Blocksworld are:

- Pickup (block)
- Putdown (block)
- Unstack (topBlock, bottomBlock)
- Stack (topBlock, bottomBlock)

To obtain visual data, we use an existing 2D simulation[1] and record its example scenarios as demonstration videos. In total, we collect 16 videos for our experiments and group them into three difficulty levels based on duration and problem complexity. The basic set contains 6 videos shorter than 11 seconds, covering 3-block problems and simple 4-block goals. The intermediate set contains 8 videos between 20 and 27 seconds, featuring 4-5 blocks and more complex goals, such as temporarily moving blocks to another stack before returning them. The hard set contains 2 videos longer than 35 seconds, both involving 8-block problems.

Overcooked is a cooperative multi-agent domain based on a cooking game, in which two or more agents must coordinate in a kitchen to prepare and deliver dishes. Unlike Blocksworld, Overcooked is not a standard benchmark in symbolic planning, but it is widely used in machine learning and reinforcement learning research. Its multi-agent nature introduces richer coordination challenges that are closer to real-world applications. In this study, we focus on two-agent scenarios across different kitchen layouts and aim to extract the high-level strategies used to complete cooking and delivery tasks. Particularly, here is the list of primitive actions:

- move (agent, destination)
- pick-up-from-pile (agent, pile, location, item)
- pick-up-from-table (agent, table, location, item)
- drop-off-from-table (agent, table, location, item)
- add-to-pot (agent, pot, location, ingredient)
- start-cook (agent, pot, location, dish)
- cook (pot, status-start, status-end)
- pour-from-pot (agent, pot, location, dish, bowl)
- drop-off-to-delivery (agent, delivery-spot, location, dish)

For visual data, we use demonstration videos collected from two human players interacting in the 2D simulated game studied from prior work (Mieczkowski et al. 2025). Although we have a larger set of human demonstration videos, in this work we use only portions of three 60-second videos, each from a different layout, due to the input token limitations of current VLMs. These three videos are chosen to represent three levels of complexity in multi-agent settings. The basic-level demonstration shows each agent trying to achieve the goal separately, without collaboration. The intermediate-level demonstration highlights collaborative methods for preparing and delivering dishes. Both the basic and intermediate levels involve only onion soup. In contrast, the hard-level demonstration includes both individual and collaborative methods for preparing dishes with onion, tomato, and mixed-ingredient soups. Although basic,

[1] https://github.com/hsu-aut/blocksworld_simulation

intermediate, and hard may not be the most precise labels for the visual-input categories in Overcooked, we use them to maintain consistency with the Blocksworld setting.

5.2 Experimental Setup

To evaluate the Vision2HTN framework, we conduct experiments across three factors:

- whether pre-defined compound tasks T_C are provided,
- the level of visual input, including no visual data and demonstrations with basic, intermediate, and hard complexity,
- the choice of VLM, including OpenAI's GPT 5.4[2], Claude Sonnet 4.6[3], and Gemini 3 Pro[4].

All three VLMs accept image inputs, but only Gemini supports native video input. To keep the evaluation consistent across models, we convert each video into a sequence of frames before providing it to the VLMs. Since the demonstration videos come from different sources, including recordings from different human players, they are converted at different frame-sampling rates, which are manually adjusted to ensure that the extracted frames capture enough information to recover the demonstrated strategies. In particular, we use 1 frame per second for Blocksworld, and 2-5 frames per second for Overcooked.

In addition to these three factors, we also compare one-shot and few-shot prompting by providing an example from the Search and Rescue domain. Since we do not observe a noticeable performance difference, we do not report this variant due to space limitations, although we retain the generated domains to enlarge our dataset. In total, this gives $2 \times 4 \times 3 \times 2 = 48$ configurations. For this preliminary study, each configuration is run three times to generate three HDDL files and obtain averaged results. Together, these settings provide a systematic setup for analyzing how task priors, visual information, and model choice influence the quality of the generated HTN outputs.

5.3 Evaluation

Quantitative evaluation of the HDDL domains generated by the Vision2HTN framework is performed using PANDA[5], an existing HDDL solver (Höller et al. 2021), which can handle both partially ordered and totally ordered problems. Each generated domain is tested on problems at three levels of complexity: basic, intermediate, and hard, corresponding to the three levels of complexity in the visual demonstration data. We record the success rate, measured by whether each generated domain can successfully produce a plan for the corresponding problems using PANDA within a 60-second runtime limit.

Qualitative analysis is used to inspect the hierarchical task networks of the generated models, especially those with poor performance. In this study, we use the graphical user interface from HDDLGym (La, Mon-Williams, and Shah 2025) to visualize the hierarchical structures of the generated HTNs. For failed domains, we also examine whether any primitive actions are missing from the generated HTNs, which may indicate that the failure is due to incompleteness in the generated domain.

[2]https://OpenAI.com/index/introducing-gpt-5-4/

[3]https://www.anthropic.com/news/Claude-sonnet-4-6

[4]https://deepmind.google/models/Gemini/pro/

[5]https://github.com/panda-planner-dev/pandaPIengine

6 Results & Discussion

Outputs of Vision2HTN, including the generated HDDL domains of Blocksworld and Overcooked, are available in our public GitHub repository. As described in Section 5.3, we evaluate our Vision2HTN framework in three factors: (1) the use of pre-defined tasks, (2) the level of visual input, and (3) the choice of VLM. For each factor, we present two mean point plots with error bars: one showing the number of tasks and methods in the generated domains, and another showing the success rate of the generated domains in solving problem sets at three levels of complexity. We also use Mann-Whitney U tests to compare the characteristics and performance across different settings. All statistically significant results are listed in Table 1 and are also marked with * or ** in the corresponding plots.

6.1 Presence of Pre-defined Tasks T_C

The characteristics and performance of the generated Blocksworld and Overcooked domains, with and without pre-defined tasks T_C, are shown in Figures 2 and 3.

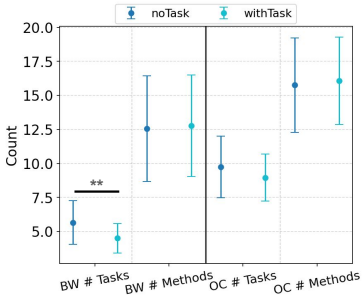

Figure 2: Task and method counts for Blocksworld (BW) and Overcooked (OC) domains when generated with and without providing tasks.

As illustrated in Figure 2, for Blocksworld, the number of generated tasks is significantly smaller when pre-defined tasks are provided than when they are not, with the p-value < 0.01. For Overcooked, the average number of generated tasks is also smaller when pre-defined tasks are provided, although the difference is not statistically significant.

Figure 3 suggests that providing pre-defined tasks has little effect in Blocksworld, but is much more helpful in Overcooked. In particular, *withTask* led to significantly higher success rates in the basic and intermediate Overcooked problem sets (both p-values < 0.01). Across domains, we also observe that performance in Blocksworld is much stronger than in Overcooked, which is expected since Overcooked is a more complex domain with richer actions and the requirement of multi-agent coordination. Overall, these results sug-

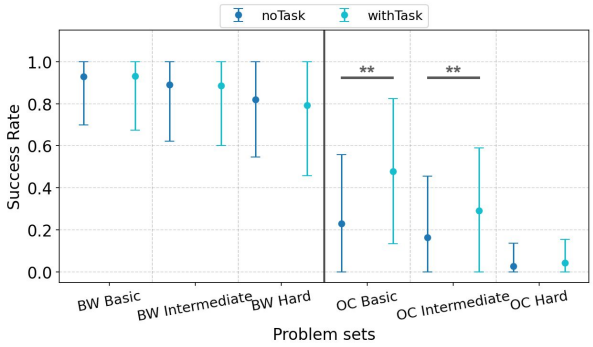

Figure 3: Success rate for Blocksworld (BW) and Overcooked (OC) domains when generated with and without providing tasks, when tested with basic, intermediate, and hard problem sets.

gest that pre-defined tasks are especially useful for guiding HTN generation in more complex domains, where the hierarchical structure is harder to infer from demonstrations alone.

6.2 Presence of Visual Data

Figures 4 and 5 demonstrate the characteristics and performance of Blocksworld and Overcooked domains when generated without visual data (labeled as 'null'), with visual data from simple problems (basic), with visual data from intermediate-level problems (intermediate), and with visual data from hard problems (hard). Figure 4 shows the number of tasks and methods are generated for each case.

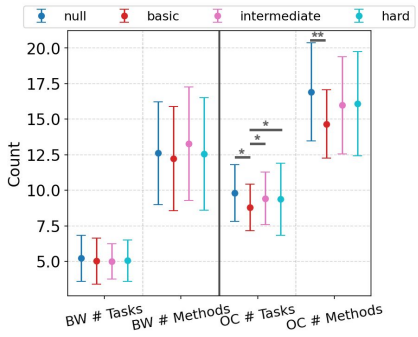

Figure 4: Task and method counts for the generated Blocksworld (BW) and Overcooked (OC) domains across four visual-input settings: no visual data (*null*) and visual data at basic, intermediate, and hard levels of complexity.

In general, the number of tasks remains relatively stable across visual-input settings in both domains. In contrast, the number of methods shows greater variation, although the pattern is not consistent across domains. In Overcooked, the domains generated without visual input have the largest average number of methods, which may suggest that visual input helps guide task decomposition and reduces unnecessary method generation. However, the no-visual setting also has

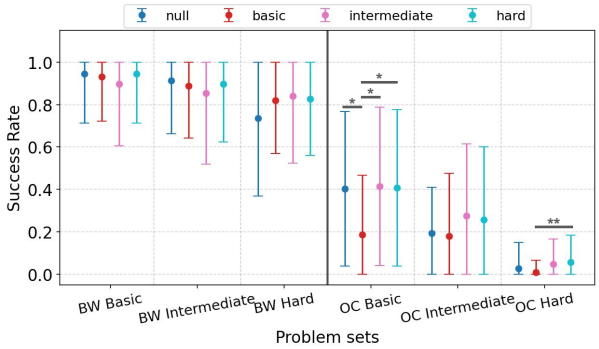

Figure 5: Success rate for Blocksworld (BW) and Overcooked (OC) domains when generated without visual data (null), with visual data of basic problems (basic), with visual data of intermediate problems (intermediate), and with visual data of hard problems (hard). They are then tested with PANDA solver with basic, intermediate, and hard problem sets.

large error bars, and the differences are statistically significant under the Mann-Whitney U test only when compared with the basic-level visual setting. Overall, this suggests that visual input may influence how the hierarchy is organized more than how many high-level tasks are introduced, but its effect on HTN size alone is not strong enough to explain performance differences.

Figure 5 shows that Blocksworld success rates remain strong across different visual-input settings, while Overcooked success rates are more sensitive and unstable. One unexpected result in Blocksworld is that domains generated without visual demonstration input achieve higher average performance than those generated with visual data. This counterintuitive outcome can be explained by the fact that Blocksworld is a popular benchmark domain, and its PDDL and HDDL representations may already appear in the pretraining data of the VLMs (Valmeekam et al. 2023). As a result, the models may rely more effectively on prior domain knowledge when no visual input is provided, whereas adding visual data, especially longer and more complex demonstrations, may introduce noise or confusion and lead to worse outputs.

In Overcooked, the results are more mixed. As noted earlier, the basic visual input demonstrates individual ways of achieving goals, the intermediate visual input demonstrates collaborative methods, and the hard-level visual input includes both. Looking at the dark blue points with error bars in Figure 5, the *null* setting, which uses no visual data, performs better on average than the visual-data settings for the basic problem set, but its performance becomes worse when handling the more complex intermediate and hard problem sets. On average, domains generated with hard-level visual data perform best on the basic problem set, which is reasonable since they may capture both individual and collaborative ways of achieving goals. For the intermediate problem set, domains generated with intermediate-level visual data perform best on average, consistent with the fact that these

Domain	Metric	Group 1	Group 2	U	p-value	Sample sizes n_1/n_2
Blocksworld	number of tasks	noTask	withTask	3732.5	0.0000**	72/72
Overcooked	basic success rate	noTask	withTask	1575.0	0.0000**	71/70
Overcooked	intermediate success rate	noTask	withTask	1735.5	0.0006**	71/70
Overcooked	num tasks	null	basic	813.0	0.0325*	36/35
Overcooked	num methods	null	basic	853.0	0.0099**	36/35
Overcooked	basic success rate	null	basic	812.5	0.0215*	36/35
Overcooked	basic success rate	basic	intermediate	434.5	0.0210*	35/35
Overcooked	basic success rate	basic	hard	437.5	0.0238*	35/35
Overcooked	hard success rate	basic	hard	525.0	0.0492*	35/35
Blocksworld	num tasks	openai	claude	1486.5	0.0116*	48/48
Blocksworld	num tasks	openai	gemini	1873.0	0.0000**	48/48
Blocksworld	num tasks	claude	gemini	1601.5	0.0007**	48/48
Blocksworld	num methods	openai	claude	632.0	0.0001**	48/48
Blocksworld	num methods	openai	gemini	2044.5	0.0000**	48/48
Blocksworld	num methods	claude	gemini	2194.0	0.0000**	48/48
Blocksworld	basic success rate	openai	claude	934.0	0.0039**	48/48
Blocksworld	basic success rate	openai	gemini	932.5	0.0037**	48/48
Blocksworld	intermediate success rate	openai	claude	800.0	0.0005**	48/48
Blocksworld	intermediate success rate	openai	gemini	816.0	0.0010**	48/48
Blocksworld	hard success rate	openai	claude	705.0	0.0003**	48/48
Blocksworld	hard success rate	openai	gemini	633.0	0.0000**	48/48
Overcooked	num tasks	openai	claude	1572.5	0.0003**	46/48
Overcooked	num tasks	openai	gemini	1824.5	0.0000**	46/47
Overcooked	num tasks	claude	gemini	1678.0	0.0000**	48/47
Overcooked	num methods	openai	claude	1614.0	0.0001**	46/48
Overcooked	num methods	openai	gemini	1807.5	0.0000**	46/47
Overcooked	num methods	claude	gemini	1548.5	0.0016**	48/47
Overcooked	basic success rate	openai	gemini	536.0	0.0000**	46/47
Overcooked	basic success rate	claude	gemini	449.0	0.0000**	48/47
Overcooked	intermediate success rate	openai	gemini	340.0	0.0000**	46/47
Overcooked	intermediate success rate	claude	gemini	358.0	0.0000**	48/47
Overcooked	hard success rate	openai	gemini	925.5	0.0333*	46/47

Table 1: Significant Mann-Whitney U test results (* $p < 0.05$, ** $p < 0.01$)

domains are constructed from purely collaborative demonstrations. For the hard problem set, the no-visual (null) and non-collaborative visual (basic) settings achieve no success, while some domains generated from collaborative or combined visual data can solve a subset of the problems. Overall, these results suggest that visual data becomes more useful in Overcooked when the target problems require collaborative structure that is difficult to infer from prior domain knowledge alone.

6.3 Choice of Vision-Language Model

We observe notable differences in both characteristics and performance of domains generated by different vision-language models. As mentioned earlier, we tested our Vison2HTN framework with the three big VLMs: OpenAI's GPT 5.4, Anthropic's Claude Sonnet 4.6, and Google's Gemini 3 Pro. They are the most recent versions of their corresponding model families at the time of this study (April 2026). The results are demonstrated in Figures 6 and 7.

For the Blocksworld domain, Figure 6 shows that Claude tends to generate more methods than the other models ($p = 0.0001$ when compared with OpenAI, and $p < 0.0001$ when compare with Gemini), and this richer hierarchy is accom-

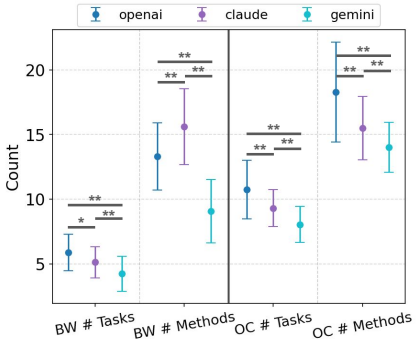

Figure 6: Task and method counts for Blocksworld (BW) and Overcooked (OC) domains when generated with GPT 5.4 (OpenAI), Claude Sonnet 4.6 (Claude), and Gemini 3 Pro (Gemini).

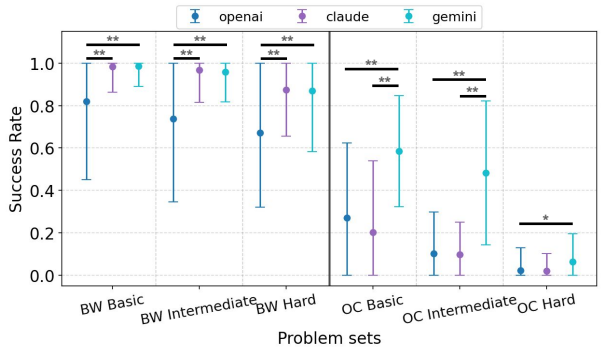

Figure 7: Success rate for Blocksworld (BW) and Overcooked (OC) domains when generated with GPT 5.4 (OpenAI), Claude Sonnet 4.6 (Claude), and Gemini 3 Pro (Gemini), when trying to solve problems in basic, intermediate, and hard problem sets with PANDA solver.

panied by stronger planning performance, especially on the harder problem sets, as shown in Figure 7.

In contrast, for the Overcooked domain, Gemini performs best overall. The Mann-Whitney U tests show that Gemini significantly outperforms OpenAI on the basic, intermediate, and hard problem sets ($p < 0.0001$, $p < 0.0001$, and $p = 0.0333$, respectively). Gemini also significantly outperforms Claude on the basic and intermediate problem sets (both have $p < 0.0001$), although the difference is not significant on the hard set. In addition, Gemini produces significantly fewer tasks and methods than both OpenAI and Claude (all have $p < 0.01$), suggesting that its generated domains may be more compact while still remaining more effective in Overcooked.

One possible explanation is that Gemini may have stronger prior familiarity with Overcooked-like visual data, since gameplay videos of cooking and coordination games are widely available online (particularly on YouTube). Overall, these results suggest that the effectiveness of a VLM for HTN generation depends not only on general model capability, but also on how well its learned priors align with the target domain.

Domain	Visual data	Self-applicability	Generality
BW	Basic	1.000 ± 0.000	0.936 ± 0.083
	Intermediate	0.738 ± 0.447	0.754 ± 0.423
	Hard	0.646 ± 0.361	0.762 ± 0.377
OC	Basic	0.182 ± 0.298	0.121 ± 0.199
	Intermediate	0.273 ± 0.389	0.231 ± 0.270
	Hard	0.061 ± 0.135	0.333 ± 0.303

Table 2: Generality and self-applicability of generated Blocksworld (BW) and Overcooked (OC) domains across problem complexity levels of visual demonstrations.

6.4 Qualitative Inspection of HTNs

In addition to quantitative analysis, we qualitatively inspect the generated HTNs to better understand the structure of the output domains, especially those with poor performance. Particularly, among the 63 generated Overcooked domains that cannot solve any problems across the three problem sets, 30 contain primitive actions that do not appear in the generated HTNs. The most common missing action is *drop-off-to-table*, which is important for collaboration because agents often need to pass items through tables, while some domains from the intermediate- and hard-visual settings miss even more critical actions such as *cook*. However, this kind of manual inspection becomes increasingly difficult in more complex domains and becomes impractical when the number of generated domains grows. Therefore, in future work, we plan to use tools such as task insertion HTN solvers (Pantčková and Barták 2025) and methods for generating missing sequences for incomplete HTNs (Muñoz Avila, Aha, and Rizzo 2025) to better identify the broken links that cause the generated domains to be incomplete.

7 Future Work

We plan to extend this work to more domains, especially those used in prior studies on generating PDDL domains and problems from visual traces. For example, ROSAME-I[6] (Xi, Gould, and Thiébaux 2024) could be a valuable source of visual data and PDDL domains for Gripper, Logistics, Towers of Hanoi, and 8-puzzle. In addition to existing 2D simulations, we also aim to incorporate 3D simulation environments and real-world visual data, such as open-source realistic-photo datasets[7].

Our current evaluation method, based on success rate with an HDDL solver (PANDA), provides only a binary signal of whether a generated domain is complete enough to solve a problem within a reasonable time. To obtain a more informative measure, we would like to apply the existing method of task insertion HTN planner (TIHTN) (Pantčková and Barták 2025) and use it to repair incomplete domains. This would allow us to estimate how far a generated domain is from a valid one by measuring the amount of insertion needed to make it solvable.

8 Conclusion

In this work, we explore the feasibility of using VLMs to learn HTN models from visual demonstrations in the Blocksworld and Overcooked domains through the Vision2HTN framework. The preliminary results suggest that visual data can support HTN learning, especially in a more complex domain such as Overcooked, where collaborative structure is harder to infer from symbolic priors alone. At the same time, the results show that the effectiveness of visual input depends on domain complexity, the form of demonstrations, and the choice of VLM. Overall, this study takes an initial step toward learning hierarchical planning models from more natural human demonstration formats, and future work can extend this approach to other domains.

[6]https://gitlab.com/xikaioliver2/ROSAME

[7]https://github.com/ibm/photorealistic-blocksworld

Acknowledgments

This work was supported in part by Lockheed Martin Corporation, and in part by the Office of Naval Research (ONR) under grant number N000142312883.

References

Athalye, A.; Kumar, N.; Silver, T.; Liang, Y.; Wang, J.; Lozano-Pérez, T.; and Kaelbling, L. P. 2026. From Pixels to Predicates: Learning Symbolic World Models via Pretrained VLMs. *IEEE Robotics and Automation Letters*.

Chen, K.; Srikanth, N. S.; Kent, D.; Ravichandar, H.; and Chernova, S. 2021. Learning Hierarchical Task Networks with Preferences from Unannotated Demonstrations. In *Proceedings of the Conference on Robot Learning*.

Dang, X.; Kudláčková, L.; and Edelkamp, S. 2025. Planning with vision-language models and a use case in robot-assisted teaching. In *AAAI 2025 Workshop on Planning in the Era of LLMs (LM4Plan)*.

Erol, K.; Hendler, J. A.; and Nau, D. S. 1995. Semantics for hierarchical task-network planning.

Fine-Morris, M.; Hsiao, V.; Smith, L. N.; Hiatt, L. M.; and Roberts, M. 2025. Leveraging LLMs for Generating Document-Informed Hierarchical Planning Models: A Proposal. In *AAAI 2025 Workshop on Planning in the Era of LLMs (LM4Plan)*.

Hao, Y.; Chen, Y.; Fan, C.; and Zhang, Y. 2026. Simulation to Rules: A Dual-VLM Framework for Formal Visual Planning. In *Proceedings of the International Conference on Learning Representations*.

Hogg, C.; Munoz-Avila, H.; and Kuter, U. 2008. HTN-MAKER: Learning HTNs with Minimal Additional Knowledge Engineering Required. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2021. The PANDA framework for hierarchical planning. *KI-Künstliche Intelligenz*.

Höller, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; and Alford, R. 2020. HDDL: An Extension to PDDL for Expressing Hierarchical Planning Problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

La, N.; Mon-Williams, R.; and Shah, J. A. 2025. HDDLGym: a tool for studying multi-agent hierarchical problems defined in HDDL with OpenAI Gym. In *Proceedings of the International Conference on Automated Planning and Scheduling*.

Langley, P. 2022. Learning hierarchical problem networks for knowledge-based planning. In *International Conference on Inductive Logic Programming*.

Langley, P. 2025. Learning Hierarchical Task Knowledge for Planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Li, R.; Nau, D.; Roberts, M.; and Fine-Morris, M. 2024. Automatically Learning HTN Methods from Landmarks. In *The International FLAIRS Conference Proceedings*.

Liang, Y.; Kumar, N.; Tang, H.; Weller, A.; Tenenbaum, J. B.; Silver, T.; Henriques, J. F.; and Ellis, K. 2025. Visualpredicator: Learning abstract world models with neuro-symbolic predicates for robot planning. In *International Conference on Learning Representations*.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D. S.; and Wilkins, D. 1998. PDDL – The Planning Domain Definition Language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

Merler, M.; Dainese, N.; Alakuijala, M.; Bonetta, G.; Ferrazzi, P.; Tian, Y.; Magnini, B.; and Marttinen, P. 2025. ViPlan: A benchmark for visual planning with symbolic predicates and vision-language models. *arXiv preprint arXiv:2505.13180*.

Mieczkowski, E.; Mon-Williams, R.; Bramley, N.; Lucas, C. G.; Velez, N.; and Griffiths, T. L. 2025. Predicting Multi-Agent Specialization via Task Parallelizability. In *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*.

Muñoz Avila, H.; Aha, D. W.; and Rizzo, P. 2025. ChatHTN: Interleaving Approximate (LLM) and Symbolic HTN Planning. In *Proceedings of the International Conference on Neuro-symbolic Systems*, Proceedings of Machine Learning Research. PMLR.

Pantčková, K.; and Barták, R. 2025. Parsing-Based Planner for Totally Ordered HTN Planning with Task Insertion. In *2025 IEEE 37th International Conference on Tools with Artificial Intelligence (ICTAI)*.

Shirai, K.; Beltran-Hernandez, C. C.; Hamaya, M.; Hashimoto, A.; Tanaka, S.; Kawaharazuka, K.; Tanaka, K.; Ushiku, Y.; and Mori, S. 2024. Vision-language interpreter for robot task planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Siburian, J.; Shirai, K.; Beltran-Hernandez, C. C.; Hamaya, M.; Görner, M.; and Hashimoto, A. 2025. Grounded Vision-Language Interpreter for Integrated Task and Motion Planning. In *CoRL 2025 Workshop on Safe and Robust Robot Learning for Operation in the Real World (SAFE-ROL)*.

Valmeekam, K.; Marquez, M.; Olmo, A.; Sreedharan, S.; and Kambhampati, S. 2023. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Proceedings of the International Conference on Neural Information Processing Systems*.

Xi, K.; Gould, S.; and Thiébaux, S. 2024. Neuro-symbolic learning of lifted action models from visual traces. In *Proceedings of the International Conference on Automated Planning and Scheduling*.

Xu, Y.; and Munoz-Avila, H. 2025. Online Learning of HTN Methods for integrated LLM-HTN Planning. In *Proceedings of the 12th Annual Conference on Advances in Cognitive Systems*.

Yang, Z.; Hedegaard, B.; Jaafar, A.; Wei, Y.; Thompson, S.; Raman, S. S.; Fu, H.; Tellex, S.; Konidaris, G.; Paulius, D.; et al. 2025. SkillWrapper: Generative Predicate Invention for Skill Abstraction. *arXiv preprint arXiv:2511.18203*.

Ye, H.; Xiao, Y.; Lu, C.; and Cai, P. 2025. UniDomain: Pretraining a Unified PDDL Domain from Real-World Demonstrations for Generalizable Robot Task Planning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Yousefi, M.; and Bercher, P. 2025. HDDL Parser: A Realtime Hierarchical Planning Language Validation Toolkit. In *ICAPS 2025 Demonstration Systems*.

Zhang, X.; Altaweel, Z.; Hayamizu, Y.; Ding, Y.; Amiri, S.; Yang, H.; Kaminski, A.; Esselink, C.; and Zhang, S. 2025. DKPROMPT: Domain Knowledge Prompting Vision-Language Models for Open-World Planning. In *AAAI 2025 Workshop on Planning in the Era of LLMs (LM4Plan)*.

Zhuo, H. H.; Muñoz Avila, H.; and Yang, Q. 2014. Learning Hierarchical Task Network Domains from Partially Observed Plan Traces. *Artificial Intelligence*.