Factored State Sampling

Matan Argaman[1], Guy Azran[1], Sarah Keren[1]

[1]Taub Faculty of Computer Science, Technion – Israel Institute of Technology
amatan@campus.technion.ac.il, guy.azran@campus.technion.ac.il, sarahk@cs.technion.ac.il

Abstract

Task planning requires accurate state estimation to achieve goals, but current methods rely on manually created, domain-specific functions that are time-consuming and struggle to adapt. Recent advances in Large Language Models (LLMs) and Vision-Language Models (VLMs) enable zero-shot semantic state estimation but often lack task-specific accuracy, leading to suboptimal plans or goal failure. To address this, we propose the Factored State Sampler (FSS), which integrates task domain knowledge to refine state estimation. While the FSS slightly reduces the micro-average AUC on state features, it substantially enhances state validity, which is a critical metric for effective task planning. The highest state validity is achieved by combining an additional neural network with the FSS, demonstrating the significant impact of our approach on enhancing generalization in task planning.

Introduction

Task planning involves searching a combinatorial state space to determine a sequence of high-level, symbolic actions (Ghallab, Nau, and Traverso 2016) (Geffner and Bonet 2013) which an agent uses to solve tasks. These plans are based on determining the internal state of the system through observations, a process known as state estimation. When an agent follows the actions which the plan dictates failure may occur and lead to unknown states. Therefore, this process is often repeated throughout the task until the agent reaches its goal. Effectively executed task plans enable intelligent agents to solve complex problems across a wide range of cutting-edge domains, including lunar rovers (Martinez Rocamora et al. 2023), autonomous vehicles (Hu et al. 2023) (Ding et al. 2020), game AI (Duarte et al. 2020), and more. To achieve generalization in task planning within these diverse visual task domains, state estimation must be automated and standardized.

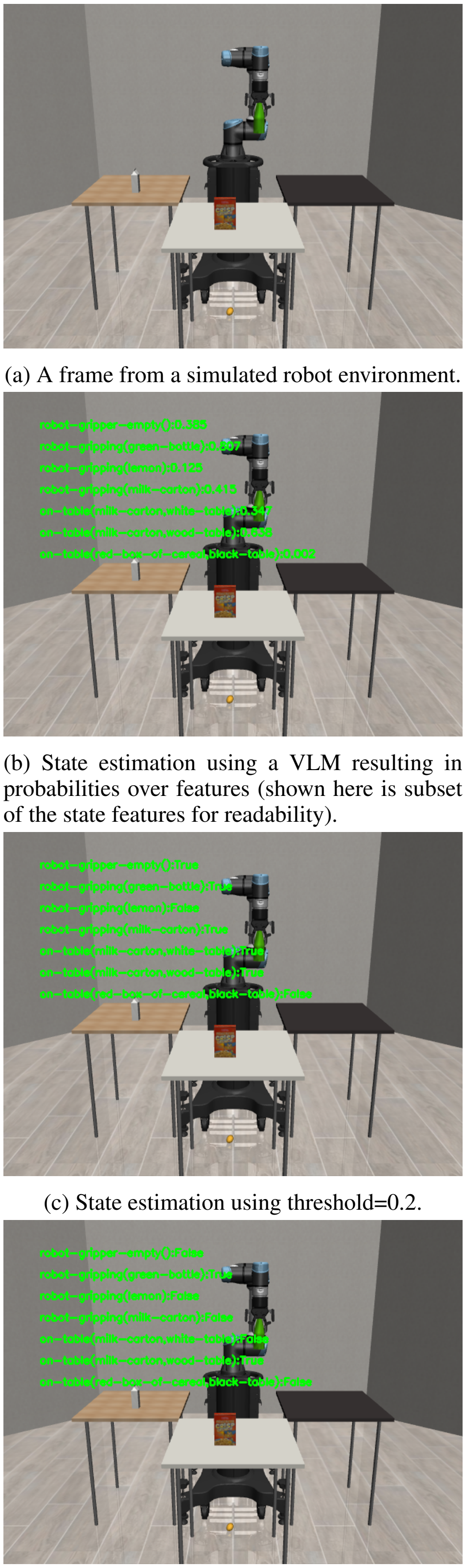
The emergence of Large Language Models (LLMs) (Radford et al. 2019) (Chen, Xiao, and Hsu 2024) and instruction-driven Visual Question Answering (VQA) foundation models (Radford et al. 2021) (Maggio et al. 2024) (Duan et al. 2024) presents a transformative opportunity to overcome these limitations. Unlike prior methods that relied on a tailored combination of computer vision tools for specialized queries, VQA models are designed to interpret visual input and answer diverse natural language questions within their training distribution. These models excel at zero-shot generalization, enabling semantic state estimation that can be directly integrated into a variety of task planning scenarios. This plug-and-play approach enhances planning flexibility and offers a scalable solution for state estimation across numerous domains.

Although these models have significantly advanced task planning performance, there remains considerable room for improvement. Accurate state estimation enables task plans that guide the agent to perform the correct actions to achieve its goal. However, inaccurate state estimation can result in an excessive number of actions where fewer would have sufficed, or worse, failure to reach the goal. When these models are used as zero-shot state estimators, their predictions are often either completely independent of the task domain, presenting a significant challenge, or fully tailored to a specific domain, making them difficult to apply to other domains. To overcome this challenge and improve performance across diverse planning tasks, we introduce a new approach called the Factored State Sampler (FSS). FSS improves state estimation accuracy by restricting states to those relevant to the task domain and enforcing mutual constraints on state features.

Background

A Task Domain

A task domain or *classical planning model* (Geffner & Bonet, 2013(Geffner and Bonet 2013)) is defined as a tuple $\Sigma = \langle S, s_0, S_G, A, T, c\rangle$ where S is a discrete and finite set of states defined over a finite feature space F (i.e., $S \ni s = (x_1, x_2, \dots, x_k)$ where $x_1, x_2, \dots, x_k$ are the features of the state), $s_0 \in S$ is a known initial state, and $S_G \subseteq S$ is a non-empty set of goal states. The function $A(s) \subseteq A$ is the set of actions applicable at state $s \in S$, T is a deterministic transition function where $T(s, a) \in S$ is the state that follows s after performing action $a \in A(s)$, and $c(a, s)$ is a positive cost of performing action a at state s. A task plan TP is a sequence of actions $TP = (a_1, a_2, \dots, a_n)$ that are applicable in order from s_0 onwards, where $\forall a_i \in TP$, $s_{t+1} = T(s_t, a_{t+1})$, and $s_n \in S_G$.


robot-gripper-empty():0.385
robot-gripping(green-bottle):0.807
robot-gripping(lemon):0.125
robot-gripping(milk-carton):0.415
on-table(milk-carton,white-table):0.347
on-table(milk-carton,wood-table):0.638
on-table(red-box-of-cereal,black-table):0.002
robot-gripper-empty():True
robot-gripping(green-bottle):True
robot-gripping(lemon):False
robot-gripping(milk-carton):True
on-table(milk-carton,white-table):True
on-table(milk-carton,wood-table):True
on-table(red-box-of-cereal,black-table):False
robot-gripper-empty():False
robot-gripping(green-bottle):True
robot-gripping(lemon):False
robot-gripping(milk-carton):False
on-table(milk-carton,white-table):False
on-table(milk-carton,wood-table):True
on-table(red-box-of-cereal,black-table):False

(a) A frame from a simulated robot environment.

(b) State estimation using a VLM resulting in probabilities over features (shown here is subset of the state features for readability).

(c) State estimation using threshold=0.2.

(d) State estimation using threshold=0.2 and Factored State Sampling. Features which cannot co-exist cancel each other by comparing probabilities, a list of canceled, canceler pairs follows: [('robot-gripper-empty()', 'robot-gripping(green-bottle)'), ('robot-gripping(milk-carton)', 'robot-gripping(green-bottle)'), ('on-table(milk-carton,white-table)', 'on-table(milk-carton,wood-table)'), ('robot-gripping(milk-carton)', 'on-table(milk-carton,wood-table)')].

Figure 1

(a) A frame from the real robot environment

Figure 2

Given a state $s_t \in S$ a deterministic policy π is defined $\pi(s_t) = a_{t+1}$.
Given a task domain Σ a task planner P is defined $P(\Sigma) \in \{\emptyset, TP\}$ where $\forall a_{t+1} \in TP, \pi(s_t) = a_t \sim \pi(\cdot|s_t)$ and $\emptyset$ denotes that no plan to reach a goal state exist.

Factored State Estimation

Let O be the observation space and let $O_t = \{o_1, o_2, ..., o_t\} \subset O$ be the set of all observations up to time t. Then given the feature space F the factored state estimator ψ is defined as probabilities over the feature space. In the binary feature space case:

$$\psi_F(O_t) \in [0,1]^{|F|}$$

Related Work

(Liu et al. 2023) introduced a framework that employs Large Language Models (LLMs) to tackle planning problems described in natural language by translating them into the Planning Domain Definition Language (PDDL) (Aeronautiques et al. 1998), a standardized language for task domain representation, and utilizing established planners to find solutions. Similarly (Guan et al. 2023) translated a problem in natural language to PDDL but assumed the correct initial symbolic state was given. However, while determining the correct symbolic state is a one-time challenge in scenarios with deterministic action outcomes, in environments with probabilistic results, this challenge is compounded by the need to repeatedly infer the symbolic state after each action. Recently, studies employing Visual Language Models (VLMs) to estimate the symbolic state of visual environments have encountered this challenge, especially in cases where uncertainty is introduced by factors like robot movements with probabilistic outcomes (Liang et al. 2024).
The aforementioned approaches, which utilize foundation models as plug-and-play state estimators, overlook critical logical information inherent in the task domain—information that we aim to demonstrate can significantly enhance multi-label classification of features.

Problem Formulation

Factored State Sampler

A factored state sampler (FSS) is a function β that, given a planner P and a factored state estimator ψ_F corresponding to task domain Σ, $\beta(\psi_F(O_t), \Sigma) \in S$ is a state sampled from distribution conditioned on Σ.

We aim to improve the multi-label classification of features. Specifically as not all combinations of values of the features lead to a valid state in the task domain σ, we look for a factored state sampler $\beta(\psi_F(O_t), \Sigma) \in S$ s.t. we maximize the micro-average AUC of state features.

Method and Evaluation

Given a factored state estimator without task domain knowledge, we use a factored state sampler which leverages the domain knowledge to provide a goal oriented state sample. This is relevant to a variety of problems. We point to 2 areas of prior knowledge found in the task domain:

- Co-occurence of features - The task domain gives us knowledge about which features cannot co-occur in a state. A simple example from blocksworld, a task domain that consists of a set of blocks that can be stacked or arranged on a table according to specific rules or goals, such as achieving a particular configuration - Two blocks cannot physically occupy the same space or be stacked on top of each other. Similarly, it is impossible for two blocks to simultaneously rest on a third block. This type of knowledge may help reduce false positives.

We present two initial baselines to evaluate our work:

- **micro-average AUC of state features** - We use a factored state sampler that relies on a co-occurrence matrix of ground truth states. In this approach, we first determine the binary values (True or False) for all features based on a threshold (1c). For every pair of features that are True but not allowed to co-occur, we identify the feature with the higher probability as the ”canceler” and the one with the lower probability as the ”canceled.” We then iterate over all ”canceler” features in order of descending probability. If a ”canceler” feature is still True, we set all its ”canceled” features to False (as shown in 1d).

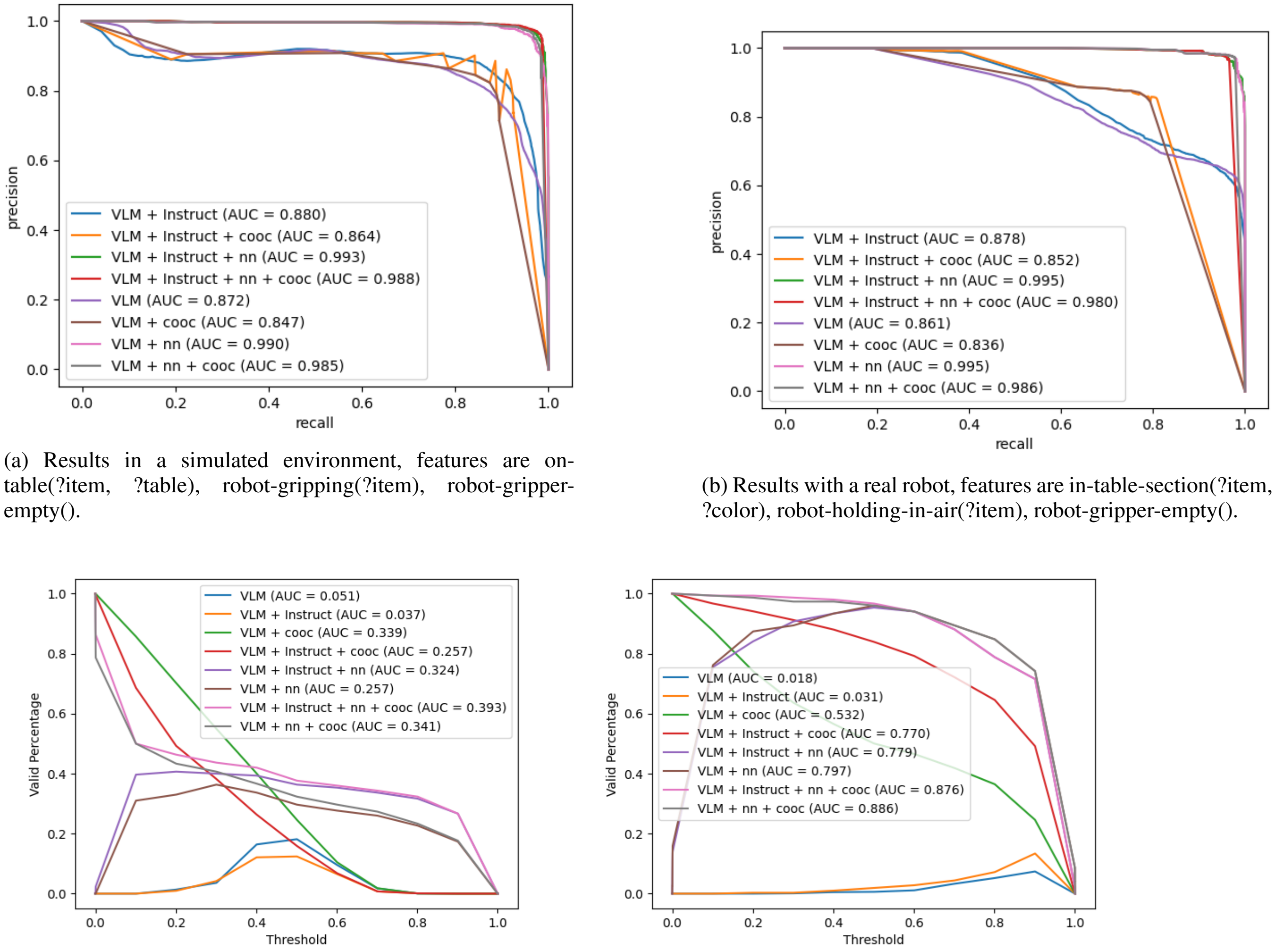
  features can exhibit dependencies spanning a distance of 3 or more, while the co-occurrence matrix only captures dependencies up to a distance of 2. Consider the blocksworld example involving three blocks: A, B, and C. A dependency of distance 2 that cannot co-occur would be: A on B and B on A. In contrast, a dependency of distance 3 that cannot co-occur would involve: A on B, B on C, and C on A. Here, A on B describes the physical relationship where block A is placed on top of block B.
- **State Validity AUC** - We use the task domain PDDL and Breadth-First Search algorithm to generate all the states of the problem; A state is considered valid if it is in this generated set and not valid otherwise.

Results

In this section, we present the results of our FSS approach as a multi-label classifier. It’s important to note that most feature labels are typically False—approximately 75% of the time. For example, when an object is on the wood table it isn’t on any other table, nor is it being gripped. This results in a scenario where a simple model that predicts all features as False for every data point can achieve near 75% accuracy. To provide a more meaningful evaluation, we use micro-average AUC of state features as our primary metric.

3a shows our results for a robot in simulated environment, where the goal is to arrange different items on specific tables. 3b shows our results for a real robot (2a) in a similar environment and goal to the simulated one. In the simulated environment we use the simulator privileged information to create the ground truth while in the real world environment we use human annotated ground truth. We use the LLaVA VLM (Liu et al. 2024) as our factored state estimator - denoted as ”VLM”. The ”VLM + instruct” denotes the same VLM, we use ”instruct” to note that we changed the action of gripping an item to include moving it to a predefined location which is in the air and not on the table. This extra movement helps solve the uncertainty in the image of whether the gripping action has succeeded or not. The ”cooc” dentoes the usage of our FSS method while the ”nn” denotes the usage of an additional neural network.

As shown in our simulated environment results 3a our FSS approach led to a decrease in micro-average AUC of 2.87% for the VLM model and 1.82% for the ”VLM + instruct” model. In the real robot environment 3b, the micro-average AUC decreased by 2.9% for the ”VLM” model and 2.96% for the VLM + instruct model. To offer additional context, we introduced an extra baseline, referred to as the ’nn’ which in both figures. This function, denoted as γ, is represented as $\gamma(\psi_F(O_t)) \in F$. In the simulated environment, γ demonstrated increases of 13.53% for the ”VLM” model and 12.84% for the ”VLM + instruct” model. In the real robot environment, γ showed increases of 15.56% for the ”VLM” model and 13.33% for the ”VLM + instruct” model. However, γ relies on the availability of ground truth data, which the FSS approach does not require. Due to this reliance on ground truth data, γ cannot be easily generalized to other task domains and must be tailored with specific ground truth data for each new domain.
When looking at the state validity results in the simulated 3c environment, and real robot 3d the ”VLM”, ”VLM + instruct” AUC values are below 5.1%. In both environments the ’nn’ and ’cooc’ improve these results by hundreds of percents. Finally, in both environments, the best model are those which include both the ’nn’ and the ’cooc’.



(a) Results in a simulated environment, features are on-table(?item, ?table), robot-gripping(?item), robot-gripper-empty().

(b) Results with a real robot, features are in-table-section(?item, ?color), robot-holding-in-air(?item), robot-gripper-empty().

(c) Results of state validity in simulated environment

(d) Results of state validity with a real robot

Figure 3: (a, b) results from simulated environment and real robot experiments showing precision-recall curves over all features for each process. micro-average AUC of state features value is written near each process's name. (c,d) Results of state validity

Conclusions

Our FSS approach did not enhance the micro-average AUC of state features, as we have not fully utilized all the information available within the task domain. Specifically, the transition function information remains untapped. Additionally, we have yet to leverage the co-occurrence data for distances of 3 and beyond. We leave the exploration of these aspects for future work.

We revisit our initial motivation for employing an FSS as a mechanism to generate valid states, enabling task planners to produce actions that guide the agent toward its goal. Enhancing the micro-average AUC significantly boosts state validity, as achieving the theoretical 100% AUC implies that all states are true and therefore valid. However, we observe that while the FSS slightly reduces the AUC, it substantially enhances validity. Notably, the best validity results are achieved when combining both the neural network and the FSS.

State validity is a crucial measure for effective task planning. Our approach demonstrates that FSS significantly enhances this metric within a framework that leverages a VLM for task planning in a generalized manner.

References

Aeronautiques, C.; Howe, A.; Knoblock, C.; McDermott, I. D.; Ram, A.; Veloso, M.; Weld, D.; Sri, D. W.; Barrett, A.; Christianson, D.; et al. 1998. Pddl— the planning domain definition language. *Technical Report, Tech. Rep.*

Chen, S.; Xiao, A.; and Hsu, D. 2024. Llm-state: Open world state representation for long-horizon task planning with large language model. *arXiv preprint arXiv:2311.17406*.

Course, K.; and Nair, P. B. 2023. State estimation of a physical system with unknown governing equations. *Nature*, 622: 261–267.

Dalmau Moreno, M.; García, N.; Gómez, V.; and Geffner, H. 2024. Combined Task and Motion Planning via Sketch Decompositions. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, 123–132.

Ding, Y.; Zhang, X.; Zhan, X.; and Zhang, S. 2020. Task-motion planning for safe and efficient urban driving. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2119–2125. IEEE.

Duan, J.; Yuan, W.; Pumacay, W.; Wang, Y. R.; Ehsani, K.; Fox, D.; and Krishna, R. 2024. Manipulate-anything: Automating real-world robots using vision-language models. *arXiv preprint arXiv:2406.18915*.

Duarte, F. F.; Lau, N.; Pereira, A.; and Reis, L. P. 2020. A survey of planning and learning in games. *Applied Sciences*, 10(13): 4529.

Geffner, H.; and Bonet, B. 2013. *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Springer International Publishing, Cham. ISBN 978-3-031-00436-0.

Ghallab, M.; Nau, D.; and Traverso, P. 2016. *Automated Planning and Acting*. Cambridge: Cambridge University Press. ISBN 978-1-107-03727-4.

Guan, L.; Valmeekam, K.; Sreedharan, S.; and Kambhampati, S. 2023. Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning. In *NeurIPS 2023*, volume arXiv:2305.14909.

Hu, Y.; Yang, J.; Chen, L.; Li, K.; Sima, C.; Zhu, X.; Chai, S.; Du, S.; Lin, T.; Wang, W.; et al. 2023. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 17853–17862.

Liang, Y.; Kumar, N.; Tang, H.; Weller, A.; Tenenbaum, J. B.; Silver, T.; Henriques, J. F.; and Ellis, K. 2024. VisualPredicator: Learning Abstract World Models with Neuro-Symbolic Predicates for Robot Planning. *arXiv preprint*, arXiv:2410.23156.

Liu, B.; Jiang, Y.; Zhang, X.; Liu, Q.; Zhang, S.; Biswas, J.; and Stone, P. 2023. LLM+P: Empowering Large Language Models with Optimal Planning Proficiency. *arXiv preprint*, arXiv:2304.11477.

Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2024. Visual instruction tuning. *Advances in neural information processing systems*, 36.

Maggio, D.; Chang, Y.; Hughes, N.; Trang, M.; Griffith, D.; Dougherty, C.; Cristofalo, E.; Schmid, L.; and Carlone, L. 2024. Clio: Real-time Task-Driven Open-Set 3D Scene Graphs. *arXiv preprint arXiv:2404.13696*.

Martinez Rocamora, B.; Kilic, C.; Tatsch, C.; Pereira, G. A. S.; and Gross, J. N. 2023. Multi-robot cooperation for lunar In-Situ resource utilization. *Frontiers in Robotics and AI*, 10.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763. PMLR.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Wang, Y.; Wei, H.; Yang, L.; Hu, B.; and Lv, C. 2023. A Review of Dynamic State Estimation for the Neighborhood System of Connected Vehicles. *SAE International Journal of Vehicle Dynamics, Stability, and NVH*, 7(3): 367–385.