Robust Visual Planning under Partial Observability and Perceptual Uncertainty

Guy Azran
Technion - Israel Institute of Technology
Haifa, Israel
guy.azran@campus.technion.ac.il

Michael Navat
Technion - Israel Institute of Technology
Haifa, Israel
michaelnavat@campus.technion.ac.il

Sarah Keren
Technion - Israel Institute of Technology
Haifa, Israel
sarahk@cs.technion.ac.il

ABSTRACT

In partially observable settings, agents must act without full knowledge of the world state and rely on uncertain state-estimation pipelines. Obtaining grounded and verifiable symbolic plans under such uncertainty remains a key challenge. Recent work has integrated Vision-Language Models (VLMs) to bridge perception and symbolic reasoning, following two main paradigms: VLM-as-planner, which maps images directly to action sequences, and VLM-as-grounder, which grounds observations into symbolic predicates used as the initial state by off-the-shelf planners. Both approaches ignore uncertainty in the planning process, compromising robustness. We introduce a third paradigm, VLM-as-probabilistic-grounder, a novel approach that captures the uncertainty of VLM predicate groundings as a probability distribution over symbolic states. This enables planning in belief space and producing robust plans under uncertainty. Experiments in simulated household robot settings show improved robustness and task success over deterministic grounding, underscoring how our approach leverages foundation models for reliable planning under uncertainty.

KEYWORDS

Legends, Myths, Folktales

ACM Reference Format:
Guy Azran, Michael Navat, and Sarah Keren. 2026. Robust Visual Planning under Partial Observability and Perceptual Uncertainty. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 11 pages.

1 INTRODUCTION

Partially observable robotics remains challenging due to the inherent uncertainty in perception and action outcomes. To overcome these challenges, many frameworks assume the Closed World Assumption [26], meaning agents have full domain knowledge, i.e., all task-related facts are known. These frameworks traditionally relied on hand-crafted solutions requiring task-specific code [7, 20, 25, 30].

Recent work has incorporated Vision-Language Models (VLMs) to infer task-relevant information from visual observations, enabling more generalizable and flexible planning systems [11, 33]. Merler et al. [19] distinguishe two paradigms for integrating VLMs into planning: VLM-as-planner, where the VLM directly generates action sequences from visual inputs, and VLM-as-grounder, where the VLM provides symbolic predicates that an off-the-shelf planner uses to compute action sequences. While symbolic grounding improves interpretability and reliability, it treats VLM predictions as deterministic, disregarding the inherent uncertainty in visual grounding. This assumption is unrealistic in domains such as household robotics, where perception is noisy and ambiguous.

(a) "Bring book to shelf" **(b) "Put bowl in sink"**

Figure 1: Observations for two tasks from the ViPlan household benchmark [19], demonstrating planning uncertainty due to (a) misleading VLM predictions (the object is mistakenly considered as being held by the gripper), and (b) partial observability (the bowl is in the drawer).

Consider the scenarios in Fig. 1. In Fig. 1a, the robot must locate and navigate to a book that is currently on a table and place it on a specific shelf, all of which are not visible in the frame. The VLM confuses the book with the radio that is behind the robot's gripper and believes it is currently holding it. This will lead to a plan that first navigates directly to the shelf where it will realize its mistake, triggering a (possibly expensive) replan. In Fig. 1b, the robot must pick up a bowl that is hidden inside a cabinet and put it in the sink. The VLM has no indication that the bowl might be inside a cabinet, so it will repeatedly plan to navigate to the bowl that it cannot see, thus never completing the task. Explicitly maintaining a belief that captures knowledge over time (e.g., the location of the bowl) and introducing robustness to uncertainty by finding plans that solve the task for multiple possible world states (e.g., book in hand and not in hand, or bowl in cabinet or not in cabinet) will enable the robot to complete the tasks with minimal replanning.

Existing approaches produce plans that fail when visual ambiguity and missing information lead to incorrect predicate assignments. To address this limitation, we use a VLM to produce predicate probabilities to maintain an explicit belief over high-level states and

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus.

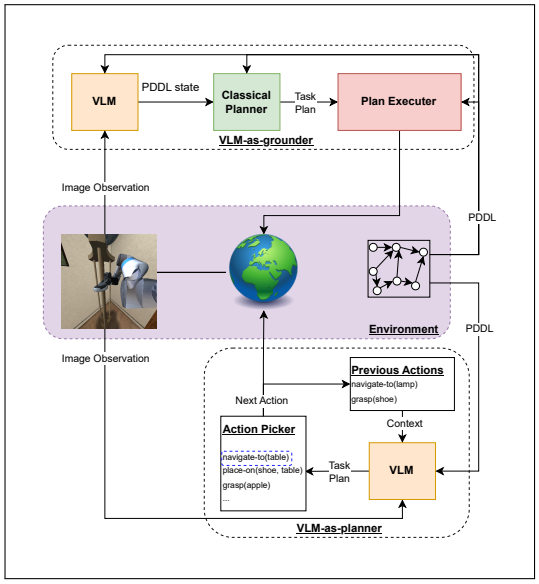

Figure 2: VLM-as-grounder (top) and VLM-as-planner (bottom)

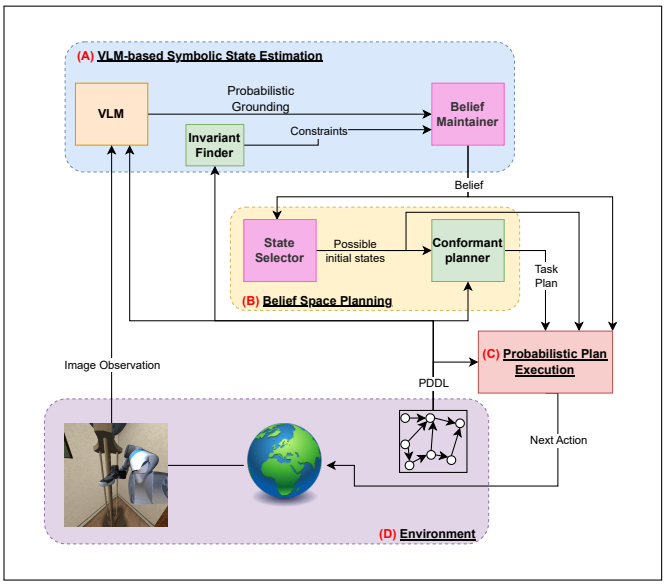

Figure 3: Robust Vision-Language Planning (RoVLaP)

framing the problem as conformant probabilistic planning (CPP) [4]. Our method, RoVLaP, leverages the VLM to guide a robust decision-making process, rather than blindly trusting its output.

The contributions of our work are as follows:

(1) We introduce VLM-as-probabilistic-grounder, a new paradigm for integrating VLMs into planning that leverages fluent-level probability distributions rather than brittle deterministic grounding or direct action generation.
(2) We formalize the Robust Visual Planning (RVP) problem, providing a principled definition of planning under partial observability and perceptual uncertainty in visual robotic domains.
(3) We propose RoVLaP, the first robust visual task planner that maintains a symbolic belief derived from VLM probabilistic grounding and compiles it into a conformant probabilistic planning problem.
(4) We develop a theoretically grounded planning and execution loop with guarantees on correctness and safety, including conditions under which VLM predictions are sufficiently accurate and useful.

We evaluate our approach on the ViPlan-HH benchmark [19] that contains multiple robot planning tasks in various home scenes. Our experiments demonstrate how RoVLaP's robust plans enable robots to solve complicated tasks under uncertainty where other approaches fail.

2 BACKGROUND AND RELATED WORK

Our approach is based on harnessing the power of pre-trained Vision-Language Models (VLMs) to answer semantic queries about visual input [15, 16, 24]. Let O be a set of possible image observations, and let $\mathcal{V}$ be a set of textual tokens called the vocabulary. A VLM is a function $\phi : O \times \mathcal{V}^* \to [0,1]^{\mathcal{V}}$ that takes as input an image observation $o \in O$ and a sequence of textual tokens (a prompt) $x = \langle v_1, \ldots, v_n \rangle \in \mathcal{V}^*$. Its output is a prediction for the next token in the sequence. By iteratively inserting the predicted token into the prompt, the VLM generates textual outputs conditioned on visual inputs.

Recent work has introduced the concept of Vision-Language-Action (VLA) models. These models leverage large-scale vision-language pretraining, often in the form of VLMs, to enable robots to understand and execute natural language instructions. VLA models have been incorporated into various architectures, including pixel-to-control policies [5, 12, 13] and within a planning loop [11, 25, 33]. In this work, we focus on the latter.

Merler et al. [19] distinguishe between two types of VLA models in a planning loop. The first is called VLM-as-planner, depicted in Fig. 2 (bottom), where a VLM is directly prompted to generate a sequence of actions to achieve a goal [5, 11, 31]. The second is called VLM-as-grounder, depicted in Fig. 2 (top), where a VLM is used to ground high-level actions into low-level executable actions [1–3, 17, 33]. In both cases, the agent does not consider the uncertainty brought on by the VLM predictions. In contrast, our work focuses on planning under uncertainty by modeling the VLM as a source of belief information, as opposed to deterministic grounding or planning.

In the VLM-as-planner approach, the VLM is prompted to generate a plan (a sequence of actions) given an image observation and a high-level task description. In the VLM-as-grounder approach, the VLM is prompted to generate true or false labels to natural language queries about the image observation, corresponding to predicates in the high-level task description, assigning value according to the VLM's response. In contrast, our approach uses the probabilities from the VLM output to define a belief over the truth

value of the predicates and plan within that space, thus accounting for the inherent uncertainty in the VLM predictions.

VLM-as-planner and VLM-as-grounder approaches use classical planning, where the agent assumes full observability of the environment and deterministic actions [9], commonly represented using the STRIPS formalism [6]. STRIPS defines a planning problem as a tuple $\langle F, I, A, G \rangle$, where F is a set of fluents representing the state of the world, I is the initial state, A is a set of actions (operators) that have preconditions that determine when they are applicable and can change the state via their effects, and $G \subseteq F$ is the goal condition. While it is common to use classical planning algorithms and replanning upon unexpected outcomes [32], this is highly ineffective in settings in which replanning is costly or even impossible. For example, a robot may need to communicate with an external computation source to plan, taking a long time due to bad connectivity or even failing when there is no connectivity.

On the other end of the spectrum, *conformant planning (CP)* [23, 28] addresses planning under partial observability by generating plans that are guaranteed to achieve the goal from any possible initial state. In conformant planning, the initial state I is replaced with a *belief set* b^I, which is a set of possible states that the agent believes it could be in. *Conformant probabilistic planning (CPP)* [4] extends this framework by replacing the belief set with a probability distribution over possible states $\beta : 2^F \to [0, 1]$, called *belief state*, or *belief*. The objective of CPP is to find a plan that achieves the goal with a probability above a specified threshold θ. Our work builds upon CPP by utilizing VLMs to define and update the belief based on visual observations, allowing for more informed planning under uncertainty in visually rich environments. We find robust plans using insights from Taig and Brafman [29] and using off-the-shelf conformant planners [18, 23, 27].

3 PROBLEM FORMULATION

We aim to construct perception–action pipelines that ground symbolic reasoning from raw visual input. These pipelines should produce satisficing task-level plans that remain effective despite sensor noise, occlusion, and partial observability.

We introduce the problem of *Robust Visual Planning (RVP)* for agents operating in complex, partially observable environments. Hereon, we focus on an embodied robotic agent, and therefore refer to it as a robot. In these settings, a robot must perceive the world through onboard sensors (e.g., cameras, depth, proprioception), infer symbolic state information from raw visual input, and select high-level actions to execute. These actions are represented as *skills*, i.e., implementations of high-level actions for the robot, that are realized by low-level motion controllers to achieve task goals.

The robot does not have direct access to the full physical state of its workspace, but only to partial and noisy observations obtained from its sensors. Its task is therefore to reason and act under perceptual uncertainty: to infer task-relevant symbolic states from limited visual information, choose appropriate high-level actions, and execute them through closed-loop control.

We formalize the problem as a tuple $\langle F, A, G, W, O, \xi \rangle$, where each component captures a layer in the perception-action hierarchy of a robotic agent:

- F, A, and G define the **symbolic task space**, following the STRIPS formalism with deterministic actions.
- W represents the **robotic workspace**, comprising the set of all feasible continuous configurations of both the robot and the manipulable objects in the scene. A workspace configuration $w \in W$ encodes the robot's joint positions, object poses, and environmental states.
- O denotes the **probabilistic perception model** which defines the space of sensory observations available to the robot (e.g., RGB-D or multi-view camera images). We model perception as a generative process $o \sim O(w)$ that captures the robot's onboard sensors and their associated noise, occlusion, and limited field of view.
- $\xi : A \times W \to \Delta W$ is the skill executor, which translates symbolic actions to low-level control policies that are executable in the workspace. Invoking $\xi(a, w)$ returns the stochastic result of applying a motion primitive or controller (e.g., for grasping or navigation) that attempts to realize the symbolic effects of $a \in A$ on current workspace configuration $w \in W$.

Note that the high-level STRIPS model is a deterministic view of the world, which defines how we expect actions to affect the workspace. However, the actual transition is handled by the non-deterministic skill executor ξ.

Our objective is to harness the capabilities of VLMs and equip robotic agents with robust planning capabilities in visually complex and uncertain environments. Assuming that for evaluation, we have access to ground-truth mapping $\sigma : W \to 2^F$ from workspace configurations to high-level STRIPS and to the initial workspace configuration w_0, we aim to generate policies that maximize the probability of reaching a goal-satisfying workspace configuration under partial observability.

4 Robust Vision-Language Planning (RoVLaP)

Our suggested approach, *Robust Vision-Language Planning (RoVLaP)*, leverages VLMs to extract probabilities about task-relevant fluents and explicitly maintain a belief for robust visual planning. We propose a framework that integrates VLM-based symbolic state estimation, probabilistic planning algorithms, and execution strategies to enable robust decision-making under perceptual uncertainty.

The RoVLaP pipeline is depicted in Fig. 3. Component (A) translates an input image observation into a symbolic belief. Here, a VLM is used to obtain a probability distribution over relevant facts about the environment. These probabilities are maintained using Bayesian belief updates. Component (B) accepts the belief as input and produces a robust plan. We select a subset of possible states for which the cumulative probability surpasses a user-specified threshold. A conformant planner generates a plan that satisfies all the states in the selected subset. Finally, component (C) handles the execution of the plan by selecting which action to take next and when to replan. We detail each component of this pipeline below.

4.1 VLM-Based Symbolic State Estimation

In the first stage of the pipeline, we infer a high-level representation of the state from an input image observation, as depicted in component (A) of Fig. 3. The common approach to using VLMs for symbolic state estimation involves prompting the model with visual

observations (e.g., images from the robot's cameras) and querying it for the truth values of task-relevant predicates in a deterministic fashion [1, 19, 33]. We also prompt the model for true-false labels of specific fluents f, but extract the internal probaibility that the VLM assigns to the next-token prediction $\phi(o, x)$ over the vocabulary, and normalize the true-false distribution to define a belief over the truth value of f.

Intuitively, we would expect the VLM to assign "uncertain" probabilities to true and false (e.g., both around 0.5) in cases where the fluent is unobservable. However, VLMs are typically trained on data where the relevant information is already in the image. Therefore, questions about objects not present in the image are considered out-of-distribution, and thus, we cannot expect them to be calibrated for this. To account for unobservable or uncertain fluents, we define a third possible fluent value `null` for cases where there is not enough evidence to confidently assign true or false.

We define *probabilistic symbolic grounding* as the extraction of a probability for each $f \in F$ in a symbolic model. Denote by x_f the prompt used to query fluent f, e.g., "Is the cabinet currently open?" for fluent `open(cabinet)`. Azran et al. [1] showed how to generate these prompts autonomously given a symbolic representation of the task domain, e.g., using Planning Domain Definition Language (PDDL). For a VLM ϕ and observation o, we calculate the probability of fluent f as follows:

(1) Extract the probabilities of the tokens "true", "false", and "null" from the VLM output. Denote

$$p_{o,f}^{true} = \phi(o, x_f)[true]$$
$$p_{o,f}^{false} = \phi(o, x_f)[false]$$
$$p_{o,f}^{null} = \phi(o, x_f)[null]$$

(2) If $p_{o,f}^{null} > p_{o,f}^{true}$ and $p_{o,f}^{null} > p_{o,f}^{false}$, set $p_{o,f} = 0.5$.
(3) Otherwise, normalize the true-false probabilities:

$$p_{o,f} = \frac{p_{o,f}^{true}}{p_{o,f}^{true} + p_{o,f}^{false}}.$$

This way we obtain a probability $p_{o,f}$ for each fluent $f \in F$ for that specific observation.

We define a *factored belief* $\beta_F = (\beta_f)_{f \in F} \in [0, 1]^F$, as a belief maintained over individual fluents. Using this representation, we perform per-fluent Bayesian belief update steps. For example, using the log-pooling update rule [8, 21] we calculate:

$$\beta_f \leftarrow \text{expit}(\text{logit}(\beta_f) + \text{logit}(p_{o,f}))$$

Note that if $p_{o,f} = 0.5$, then $\text{logit}(p_{o,f}) = 0$, and thus the belief about fluent f remains unchanged, as expected for an uninformative observation.

To obtain a belief over the full symbolic state, we assume conditional independence between fluents, subject to constraints C, e.g., mutual exclusion and co-dependence defined by the task domain. States that violate C are assigned zero probability, and the rest are normalized accordingly.

Denote by $s(f)$ the assignment of fluent f in state s, i.e., $s(f) = \mathbb{1}_{f \in s}$. The belief is calculated as follows:

$$\tilde{\beta}(s) = \prod_{f \in F} \beta_f^{s(f)} (1 - \beta_f)^{1 - s(f)} \tag{1}$$

$$\beta(s) = \begin{cases} \frac{\tilde{\beta}(s)}{Z} & \text{if } s \text{ satisfies } C \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $Z \leq 1$ is a normalization constant. We call $\tilde{\beta}$ the *unconstrained belief*, as it does not take into account the constraints in C. Note that for bto be well-defined, we must assume there exists at least one state s that satisfies C with a non-zero $\tilde{\beta}(s)$.

To extract constraints set C, we use the fast downward planning system's finite domain representation [10]. This reveals several constraints, e.g., mutually exclusive fluent groups, exposed in a dedicated invariant discovery stage. It is also possible to use the graphs generated by this invariant finder to calculate Z in polynomial time. However, ignoring this term, we can view our belief as a lower bound on the actual belief. This will be enough to obtain a state space that guarantees a desired threshold probability of success in the underlying CPP problem (see next section).

4.2 Conformant Probabilistic Planning with VLM-based Belief

Next in the pipeline is component (B) in Fig. 3, which generates a robust plan that satisfies the goal with high probability according to our maintained belief.

The CP paradigm addresses planning under initial state uncertainty. This aligns the two sources of uncertainty in our setting: perceptual uncertainty from the translation from observations to symbolic states, and partial observability in a single observation that may not fully reveal the true symbolic state. When the agent receives an observation, we want to define a CP problem that reflects the current belief.

Let $\langle F, \beta, A, G, \theta \rangle$ CPP problem. Taig and Brafman [29] prove that solving this problem equivalent to solving a CP problem $\langle F, b^I, A, G \rangle$ such that the probability that the initial state is in b^I is greater than θ, i.e., $\sum_{s \in b^I} \beta(s) \geq \theta$. However, their approach used a planner to find a b^I with the cheapest solution. We want the robot to act on the best interpretation of the visual scene, and so we propose finding the most likely subset of states that satisfy the probability threshold.

Definition 1. *Let β_F be a factored belief, $\theta \in [0, 1]$ be a probability threshold, and C be a set of constraints over F. The Most Likely Subset of States (MLSS), denoted by $MLSS(\beta_F, \theta, C)$, is a subset of states $b^I \subseteq 2^F$ of minimal cardinality such that $\sum_{s \in b^I} \beta(s) \geq \theta$.*

In a CPP problem, the Most Likely Subset of States (MLSS) is the smallest subset of states $b^I \subseteq 2^F$ such that $\sum_{s \in b^I} \beta(s) \geq \theta$.

Algorithm 1 describes our approach to finding an MLSS by performing a search over boolean fluent value flips. In lines 1-4, we initialize the search with the most likely state s_0 and set up a max-heap containing only this state. In line 7, we extract the most likely state from the heap, based on its unconstrained belief $\tilde{\beta}$, i.e., the belief before applying constraints and normalizing (computed via Eq. (1)). The extracted state is added to the output belief set if it

Algorithm 1 Most Likely Subset of States

Require: $\beta_F : F \to [0,1]$, $\theta \in [0,1]$, $C-$ set of constraints.
Ensure: $MLSS(\beta_F, \theta, C)$

1: $s_0 \leftarrow \mathbb{1}[\beta_f > 0.5] \quad \forall f \in F$
2: $Q \leftarrow$ Empty max-heap
3: Q.Insert$(s_0, \tilde{\beta}(s_0))$ $\quad$ {$\tilde{\beta}$ from Eq. (1)}
4: Visited $\leftarrow \{s_0\}$, $b \leftarrow \emptyset$
5: $p \leftarrow 0$
6: **while** $p < \theta$ **and** $Q \neq \emptyset$ **do**
7: $\quad s \leftarrow Q$.ExtractMax()
8: $\quad$ **if** $s \models C$ **then**
9: $\quad\quad b \leftarrow b \cup \{s\}$
10: $\quad\quad p \leftarrow p + \beta(s)$ $\quad$ {β from Eq. (2)}
11: $\quad$ **end if**
12: $\quad$ **for** $f \in F$ **do**
13: $\quad\quad s' \leftarrow$ Flip(s, f)
14: $\quad\quad$ **if** $s' \notin$ Visited **then**
15: $\quad\quad\quad$ Visited $\leftarrow$ Visited $\cup \{s'\}$
16: $\quad\quad\quad Q$.Insert$(s', \tilde{\beta}(s'))$
17: $\quad\quad$ **end if**
18: $\quad$ **end for**
19: **end while**
20: **return** b

satisfies the constraints (line 9). The loop starting at line 12 explores neighboring states by flipping each fluent in turn, adding unvisited neighbors to the heap (line 16). The termination condition (line 6) checks whether the cumulative probability of the selected states meets or exceeds θ. The key insight behind this algorithm is that although states are extracted from the heap according to the unconstrained belief $\tilde{\beta}$, they are guaranteed to be in non-increasing order of the belief β.

Theorem 1. *In Algorithm 1, when a constraint-satisfying state s is extracted from the heap Q, then for all $s' \notin b$ it holds that $\beta(s) \geq \beta(s')$, where β is the belief computed by Eq.* (2).

The proof shows this by asserting that states are extracted from the heap in non-increasing order of their unconstrained belief $\tilde{\beta}$ across the entire state space 2^F. The full proof can be found in the supplementary materials.

In the worst case, the number of explored states is exponential in $|F|$. Even if we bound the minimal subset size to k, there might still be a large number of constraint-violating states that need to be explored before finding k valid states. In the absence of constraints, we can provide a tighter bound on the runtime, hoping that in practice we encounter a manageable number of constraint-violating states.

Proposition 1. *If $C = \emptyset$, Algorithm 1 runs in $O(|F|k \log |F|k)$ time with $O(|F|^2 k)$ bits, where k is the number of states in the MLSS.*

With this, we can efficiently generate the CPP problem that arises from the VLM-based belief state estimation. By characterizing the accuracy of a VLM by its cumulative error over all fluents, we can guarantee that if the VLM's per-fluent predictions are *sufficiently* accurate, the true state will be included in the MLSS. This enables the user to define the bounds on sufficiency and accuracy informatively. As a first step, we prove a property on the factored belief that is necessary to guarantee the goal is achievable by the partially informed agent.

Definition 2 (Cumulative factored belief error). *Let $s \in 2^F$ be some high-level state of the world, and let $\beta_F = (\beta_f)_{f \in F}$ be a factored belief. The* cumulative factored belief error *function is defined as:*

$$\delta(\beta_F, s) = \sum_{f \in F} |\beta_f - s(f)|$$

Theorem 2. *Let δ be the cumulative factored belief error function. Then for all $s \in 2^F$ and $\beta_F \in [0,1]^F$, if $\delta(\beta_F, s) < \theta$, then s is in the MLSS.*

The proof is based on the Weierstrass Product inequality to bound the probability of states in the belief set. The full proof is provided in the supplementary material. When the true current state s^* is in the MLSS, then a conformant plan on the MLSS also satisfies the goal from the true state. We now show that this situation can be achieved in a finite number of steps, given that the VLM's predictions are not misleading, i.e., monotonically non-decreasing with a guarantee of eventually increasing.

Definition 3 (Weakly calibrated VLM). *For fluent probability p_f and true state fluent assignment $s^*(f)$, define $p_{correct}$ as the probability of the true assignment, i.e., p_f if $s^*(f) = 1$ and $1 - p_f$ otherwise, with complement $p_{incorrect} = 1 - p_{correct}$. A VLM is* weakly calibrated *if there exists $\gamma > 0$ such that for every observation $o \in O$ and fluent $f \in F$, the probability p_f derived for fluent f is at least γ closer to the correct prediction than the incorrect one, i.e.,*

$$p_{correct} - p_{incorrect} \geq \gamma$$

.

Definition 4 (Minimum visibility rate). *The* minimum visibility rate, *denoted ρ, is the frequency with which the rarest fluent is observed, where the rarest fluent is defined as a minimizer in*

$$\arg\min_{f \in F} \Pr(p_{o,f}^{null} > \max\{p_{o,f}^{true}, p_{o,f}^{false}\})$$

Corollary 1. *Let $\rho > 0$ be the minimum visibility rate. If the VLM is weakly calibrated with error margin γ, then there exists a finite number of steps T after which a conformant plan for the MLSS according to β_F is satisficing from the true current state.*

In the proof, we find a lower bound on the current log odds in the belief that grows over time, and push it above θ. What this basically means is that the VLM does not have to be perfect, but only better than a random guess, and that guarantees that eventually we will generate a satisficing plan for the real world state. The full proof is in the appendix.

Note that assuming weak calibration and minimum visibility is much weaker than the assumptions of prior work using the VLM-as-grounder paradigm, which typically require perfect predictions for all fluents all the time. Furthermore, both assumptions can be empirically verified for a given VLM and task domain, and the parameters γ and ρ can be used to informatively calculate the number of steps required to guarantee a conformant plan for the true state, as shown in the following corollary.

Of course, when using a conformant planner, there is an inherent tradeoff between robustness and completeness. The CPP component of RoVLaP allows the user to iteratively adjust this by setting the calibrating the success probability threshold θ if a plan is not found.

4.3 Executing Conformant Probabilistic Plans

We aim to support robotic settings, wherein executing a plan may lead to unexpected outcomes due to perceptual uncertainty, partial observability, and action failure. As such, part of the robustness of our policy relies on the ability of the executor to detect failure and inconsistency. A replan may be triggered when new observations indicate that the current belief is inconsistent with the belief set used for planning, or the action execution failed. In the classical case, initiating replanning is straightforward: when an action cannot be executed, or its observed outcome does not match the expected state, the agent replans from the new observed state [32]. In the conformant case, there are a few more considerations to take into account.

To handle replanning, we introduce a novel conformant plan executor, handled in component (C) of Fig. 3. In our setting, we need to determine when the current plan no longer aligns with the belief derived from new observations. Our execution component introduces a belief-consistency-based monitoring criterion, a probabilistic monitoring scheme tailored for VLM uncertainty. VLMs can drastically change their predictions from one observation to another, so the standard "expected state vs observed state" criterion can lead to constant replanning. As such, we propose multiple replanning triggers based on this misalignment: unsafe action, improbable plan, and plan exhaustion. We focus on belief drift via the "improbable plan trigger", which requires the VLM to output predictions that are persistently inconsistent with the belief and with great confidence. The "unsafe" trigger protects against "hallucinated certainty", where a VLM might initially be confident but wavers as the robot approaches. Finally, the "plan exhaustion trigger" ensures that if the plan is not successful after execution, we replan to correct for any misinterpretations of the environment. To our knowledge, this is the first execution framework that explicitly reasons about VLM-induced belief drift rather than single-step perceptual inconsistency. The full execution loop is outlined in Algorithm 2.

An unsafe action is one for which the planner did not verify the preconditions. As inspired by Shani and Brafman [27], we define an action as safe if its preconditions hold for all states in the MLSS. When an action is unsafe, we replan from the current belief.

An improbable plan is one whose probability of success is below a threshold θ given the current belief. As a proxy for this value, we calculate the probability of the belief set b^I used by the conformant planner, after executing all actions taken thus far, relative to the current belief. If this value drops below θ, it is possible that the plan's success probability is also below θ, and so we replan.

Finally, plan exhaustion occurs when all actions in the current plan have been executed without achieving the goal condition threshold. This indicates that the plan was insufficient to reach the goal from the current belief, prompting a replanning step. This is especially important in our setting, as perceptual uncertainty may lead to misinterpretations of the environment that require new plans to correct.

Algorithm 2 CPP Execution Loop

Require: $\langle F, A, G, W, O, \xi \rangle$ - an RVP problem; θ - plan success probability threshold; β^I - initial belief; o - initial observation

$C \leftarrow$ ExtractConstraints(F, A)
$\beta \leftarrow$ VLM-State-Estimator(o, ϕ, β^I, C)
while $\beta(G) < \theta$ **do**
  $b^I \leftarrow$ MLSS(β, θ)
  $\Pi \leftarrow$ Conformant-Planner(F, b^I, A, G, θ)
  for $a \in \Pi$ **do**
    if Unsafe(a, b^I) **then**
      break {Replan}
    end if
    Execute(a, ξ)
    $o \leftarrow$ SensorReading()
    $\beta \leftarrow$ VLM-State-Estimator(o, ϕ, β, C)
    if Improbable$(\Pi_{\text{remaining}}, \beta)$ **then**
      break {Replan}
    end if
  end for
end while

The resulting policy from Algorithm 2, which we will denote as π_{cpp}, guarantees robustness based on the accuracy of our perceptual belief, according to Theorem 2. With this replanning strategy, we can also provide safety guarantees compared to a VLM-as-grounder policy [19], denoted π_{det}, that always plans from the most probable state.

Theorem 3. *Let unsafe(π) denote the event that policy π executes an unsafe action. Then,*

$$\Pr(\textit{unsafe}(\pi_{det})) \geq \Pr(\textit{unsafe}(\pi_{cpp})) + (\theta - \beta(s_{max}))$$

where s_{max} is a maximizer of β.

Theorem 3 states that our CPP replanning policy is at least as safe as the VLM-as-grounder policy as long as the planning threshold is greater than the probability of the most likely state (full proof in the supplementary material). We see that the safety advantage of our method grows exponentially with the entropy of the VLM-induced belief. Although $\theta - \beta(s_{max})$ can be negative if $\beta(s_{max}) > \theta$, in practice Algorithm 1 will select only this maximizer, making the resulting plan equivalent to that of the VLM-as-grounder policy.

The computational requirements of Algorithm 2 are divided into three main components: querying the VLM to obtain per-fluent probabilities, compiling the resulting belief into a CP problem via MLSS (Algorithm 1), and solving the conformant problem using an off-the-shelf planner. The VLM is queried once per grounded fluent, so this stage is linear in the number of fluents |F| (in number of calls), and in practice dominates wall-clock time because VLM calls are expensive. The MLSS algorithm has a worst-case exponential time complexity in the number of fluents, but as shown in Proposition 1, this is manageable when the number of constraint-violating states is limited. Finally, CP itself is worst-case exponentially hard, meaning RoVLaP inherits the conformant planners' theoretical complexity.

5 USE CASE: ROBUST VISUAL PLANNING FOR A SIMULATED HOUSEHOLD ROBOT

Having formulated Robust Visual Planning and offering an approach for robust planning and plan execution, we now turn to demonstrating its relevance and effectiveness in a simulated robotic setting. The objective is to demonstrate real-world utility via representative use-cases, serving as supporting evidence for the conceptual and theoretical contributions rather than a comprehensive empirical evaluation. The scenarios presented here demonstrate failure modes of current VLM-based planning approaches in intentionally simple but representative settings that RoVLaP handles gracefully.

5.1 Setting

We use the ViPlan-HH benchmark [19]. This is a household robotics domain based on tasks from the iGibson environment [14]. In this domain, demonstrated in Fig. 1, the robot is a mobile manipulator equipped with an RGB camera and a single arm capable of grasping one object at a time. The house consists of a variety of movable objects, such as books, cups, and plates, as well as containers like cabinets and refrigerators. The robot is tasked with manipulating household objects such as opening cabinets, placing items on shelves, and arranging furniture to achieve a desired room configuration.

The domain and different problems are associated with a PDDL representation that captures the objects in the environment, including movable objects (e.g. `hardback_1`) and non-movable objects (e.g.,`table_1`), the relations between the objects (e.g., `(ontop hardback_1 table_1)`), and the high-level actions that can be performed, for example:

```
(:action place-on
    :parameters
    (?m - movable ?o2 - object)
    :precondition (and(reachable ?o2))
    :effect (and (when (holding ?m)
            (and
                (ontop ?m ?o2)
                (not (holding ?m))))))
```

The actions include navigating to certain locations, picking up objects, placing objects, and opening or closing containers. Actions that involve motion are implemented using probabilistic motion planners that can fail due to kinematic constraints, collisions, or other factors.

The observations in this domain are RGB images from the robot's point of view (see examples in Fig. 1). This means that the workspace is only partially observable, as the robot can only see a portion of the environment at any given time, making it a true partially observable planning domain. Due to the partial observability, the robot must therefore reason about its uncertainty and replan as it gathers more information through its observations.

5.2 Setup

We compare our approach to two common approaches for visual planning depicted in Fig. 2. All three approaches receive image observations from the environment, which are then processed by a VLM along with additional context derived from the PDDL-based description. All methods use GPT4.1 [22] as their VLM. GPT is invoked via the OpenAI API, which is given the image observation and a prompt, and returns the text response (for VLM-as-planner) or log probabilities (for RoVLaP and VLM-as-grounder) used in the descision-making process. The prompts used for each method are detailed in the supplementary material.

We compare three approaches, each representing one of the three paradigms for VLA in a planning loop, depicted in Figs. 2 and 3.

5.2.1 VLM-as-planner. A VLM-as-planner implemented as the Vision Language (ViLa) Planning architecture [11]. The VLM outputs a task plan in a specific format directly. The first action in that plan is taken as the agent's next action. A list of all previously selected actions is added to the VLM's context to provide the system with some memory of past interactions with the environment.

5.2.2 VLM-as-grounder. A VLM-as-grounder implemented as in Merler et al. [19] as a variation of Semantic Symbolic State Estimation (S3E) [1]. The VLM outputs a grounded PDDL state, obtained by asking a series of yes-no questions, each corresponding to a single grounded predicate. The grounded state is passed to the fast-downward planner [10] to produce a task plan that is passed to the executor. The executor uses the VLM to monitor execution by checking preconditions and effects of the new observations. When these become inconsistent with the current action, replanning is triggered.

Importantly, in the implementation by Merler et al. [19], to bypass partial observability, the grounded state is supplemented with privileged information from the simulation in the form of ground-truth assignments for predicates containing objects that are not visible. To increase the fidelity and reliability of our approach, we do not assume access to this information.

5.2.3 RoVLaP. As described above, and depicted in Fig. 3, in our approach, we extract probabilities from the VLM for each grounded predicate via the logits from the final layer of the network. We use these probabilities to update a per-predicate belief using logarithmic opinion pooling belief update [21]. Using Algorithm 1, a subset of states whose probability is at least some threshold probability is selected for planning. The state selection process uses constraints from the fast-downward invariant finder [10] to filter out impossible states. We then find a conformant plan for the selected subset of states, if one exists, using the CPOR planner [18] with a fast-downward internal planner. If no plan is found, we perform a binary search to find the largest threshold probability for which a plan exists. If a plan is found, it is executed, updating the belief at every step.

5.3 Analysis

We qualitatively examined scenarios from the ViPlan-HH benchmark in which RoVLaP mitigates perception-induced errors that cause failures in the ViLa baseline. In both examples below, RoVLaP succeeds by maintaining a belief over possible states and planning for ambiguity, rather than committing to a single VLM prediction. The PDDL domain and problem files for these examples are included in the supplementary material.

In the *cleaning out drawers* task, the robot must place a bowl into the sink, but the bowl is initially out of view and located inside a closed cabinet (see Fig. 1b). ViLa consistently mispredicts the initial state, assuming the bowl is directly reachable. As a result, it generates the following plan:

```
navigate-to      bowl_1
grasp            bowl_1
navigate-to      sink_1
place-on         bowl_1 sink_1
```

This plan inevitably fails when the bowl is inside a cabinet, since grasping it requires opening the cabinet first. In contrast, RoVLaP assigns non-zero probability to the bowl being hidden and thus produces a belief-robust plan:

```
navigate-to      cabinet_1
open-container   cabinet_1
navigate-to      bowl_1
grasp            bowl_1
navigate-to      sink_1
place-on         bowl_1 sink_1
```

This plan is valid whether or not the bowl is actually inside the cabinet. By choosing a robust plan that includes opening the cabinet, RoVLaP avoids premature commitment and expands the set of possible successful initial states, reducing the need for replanning.

In the *sorting books* task, the robot must place a hardback book onto a shelf. The initial camera image can be misleading: objects near the gripper may appear as if they are being held (Fig. 1a). Under these conditions, the VLM-as-grounder (if we remove its object detection capabilities) baseline often incorrectly interprets the scene, e.g., concludes that the robot is already holding the hardback. This misinterpretation causes the robot to generate the following plan:

```
navigate-to      shelf_1
place-on         hardback_1 shelf_1
```

Because the robot is not actually holding the `hardback_1`, this plan fails. RoVLaP, however, maintains uncertainty over the `holding` predicate and generates a plan that is feasible under both hypotheses (holding vs. not holding):

```
place-on         hardback_1 shelf_1
navigate-to      hardback_1
grasp            hardback_1
place-on         hardback_1 shelf_1
```

This plan ensures task success even when the robot begins empty-handed, avoiding the brittle dependence on a single incorrect perceptual judgment.

6 CONCLUSION

To enable robotic agents to operate effectively in real-world, partially observable environments characterized by limited and uncertain perceptual information, we propose a framework for robust planning under perceptual uncertainty. The approach employs Vision-Language Model (VLM) to derive a probabilistic, factored representation of the current state, which is used to update the agent's belief. This belief representation supports the synthesis of robust task plans using off-the-shelf planners.

As a natural extension of this work, we will investigate the incorporation of active sensing into the planning process, enabling agents to reason explicitly about the informational value of sensing actions and to integrate perceptual capabilities into their sequential decision-making. In addition, we plan to embed the proposed methodology within embodied robotic systems and evaluate its effectiveness across a suite of complex task-and-motion planning (TAMP) domains, thereby assessing its practical utility in real-world robotic operation.

REFERENCES

[1] Guy Azran, Yuval Goshen, Kai Yuan, and Sarah Keren. 2025. S3E: Semantic Symbolic State Estimation With Vision-Language Foundation Models. In *AAAI 2025 Workshop LM4Plan*.

[2] Siwei Chen, Anxing Xiao, and David Hsu. 2024. LLM-State: Open World State Representation for Long-horizon Task Planning with Large Language Model. https://doi.org/10.48550/arXiv.2311.17406 arXiv:2311.17406 [cs]

[3] Yan Ding, Xiaohan Zhang, Saeid Amiri, Nieqing Cao, Hao Yang, Chad Esselink, and Shiqi Zhang. 2024. Robot Task Planning and Situation Handling in Open Worlds. https://doi.org/10.48550/arXiv.2210.01287 arXiv:2210.01287 [cs]

[4] Carmel Domshlak and Jörg Hoffmann. 2006. Fast Probabilistic Planning through Weighted Model Counting. In *Proceedings of the Sixteenth International Conference on International Conference on Automated Planning and Scheduling (ICAPS'06)*. AAAI Press, Cumbria, UK, 243–252.

[5] Jiafei Duan, Wentao Yuan, Wilbert Pumacay, Yi Ru Wang, Kiana Ehsani, Dieter Fox, and Ranjay Krishna. 2024. Manipulate-Anything: Automating Real-World Robots Using Vision-Language Models. In *8th Annual Conference on Robot Learning*.

[6] Richard E. Fikes and Nils J. Nilsson. 1971. Strips: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2, 3 (Dec. 1971), 189–208. https://doi.org/10.1016/0004-3702(71)90010-5

[7] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2020. PDDLStream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning. arXiv:1802.08705 [cs]

[8] Christian Genest and James V. Zidek. 1986. Combining Probability Distributions: A Critique and an Annotated Bibliography. *Statist. Sci.* 1, 1 (1986), 114–135. arXiv:2245510

[9] Malik Ghallab, Dana S. Nau, and Paolo Traverso. 2004. *Automated Planning: Theory and Practice*. Elsevier/Morgan Kaufmann, Amsterdam Boston.

[10] M. Helmert. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research* 26 (July 2006), 191–246. https://doi.org/10.1613/jair.1705

[11] Yingdong Hu, Fanqi Lin, Tong Zhang, Li Yi, and Yang Gao. 2023. Look Before You Leap: Unveiling the Power of GPT-4V in Robotic Vision-Language Planning. https://doi.org/10.48550/arXiv.2311.17842 arXiv:2311.17842 [cs]

[12] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. 2023. VIMA: General Robot Manipulation with Multimodal Prompts. In *International Conference on Learning Representations (ICLR)*. arXiv. https://doi.org/10.48550/ARXIV.2210.03094

[13] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. 2024. OpenVLA: An Open-Source Vision-Language-Action Model. arXiv:2406.09246

[14] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Elliott Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. 2022. iGibson 2.0: Object-centric Simulation for Robot Learning of Everyday Household Tasks. In *Proceedings of the 5th Conference on Robot Learning (Proceedings of Machine Learning Research, Vol. 164)*, Aleksandra Faust, David Hsu, and Gerhard Neumann (Eds.). PMLR, 455–465.

[15] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. VisualBERT: A Simple and Performant Baseline for Vision and Language. https://doi.org/10.48550/arXiv.1908.03557 arXiv:1908.03557 [cs]

[16] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, Kai-Wei Chang, and Jianfeng Gao. 2022. Grounded Language-Image Pre-training. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10955–10965. https://doi.org/10.1109/CVPR52688.2022.01069

[17] Yichao Liang, Nishanth Kumar, Hao Tang, Adrian Weller, Joshua B. Tenenbaum, Tom Silver, Joao F. Henriques, and Kevin Ellis. 2024. VisualPredicator: Learning Abstract World Models with Neuro-Symbolic Predicates for Robot Planning. In *The Thirteenth International Conference on Learning Representations*.

[18] Shlomi Maliah, Radimir Komarnitski, and Guy Shani. 2022. Computing Contingent Plan Graphs Using Online Planning. *ACM Trans. Auton. Adapt. Syst.* 16, 1 (Jan. 2022), 1:1–1:30. https://doi.org/10.1145/3488903
[19] Matteo Merler, Nicola Dainese, Minttu Alakuijala, Giovanni Bonetta, Pietro Ferrazzi, Yu Tian, Bernardo Magnini, and Pekka Marttinen. 2025. ViPlan: A Benchmark for Visual Planning with Symbolic Predicates and Vision-Language Models. https://doi.org/10.48550/arXiv.2505.13180 arXiv:2505.13180 [cs]
[20] Magí Dalmau Moreno, Néstor García, Vicenç Gómez, and Héctor Geffner. 2024. Combined Task and Motion Planning via Sketch Decompositions. *Proceedings of the International Conference on Automated Planning and Scheduling* 34 (May 2024), 123–132. https://doi.org/10.1609/icaps.v34i1.31468
[21] Eric Neyman and Tim Roughgarden. 2023. No-Regret Learning with Unbounded Losses: The Case of Logarithmic Pooling. https://doi.org/10.48550/arXiv.2202.11219 arXiv:2202.11219 [cs]
[22] OpenAI. 2023. GPT-4V(Ision) System Card. https://openai.com/index/gpt-4v-system-card/.
[23] H. Palacios and H. Geffner. 2009. Compiling Uncertainty Away in Conformant Planning Problems with Bounded Width. *Journal of Artificial Intelligence Research* 35 (Aug. 2009), 623–675. https://doi.org/10.1613/jair.2708
[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 8748–8763.
[25] Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. 2023. SayPlan: Grounding Large Language Models Using 3D Scene Graphs for Scalable Robot Task Planning. In *7th Annual Conference on Robot Learning*.
[26] Raymond Reiter. 1981. ON CLOSED WORLD DATA BASES. In *Readings in Artificial Intelligence*, Bonnie Lynn Webber and Nils J. Nilsson (Eds.). Morgan Kaufmann, 119–140. https://doi.org/10.1016/B978-0-934613-03-3.50014-3
[27] Guy Shani and Ronen I. Brafman. 2011. Replanning in Domains with Partial Information and Sensing Actions. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three (IJCAI'11)*. AAAI Press, Barcelona, Catalonia, Spain, 2021–2026.
[28] David E. Smith and Daniel S. Weld. 1998. Conformant Graphplan. In *AAAI/IAAI*.
[29] Ran Taig and Ronen I. Brafman. 2013. Compiling Conformant Probabilistic Planning Problems into Classical Planning. *Proceedings of the International Conference on Automated Planning and Scheduling* 23 (June 2013), 197–205. https://doi.org/10.1609/icaps.v23i1.13540
[30] Or Wertheim, Dan R. Suissa, and Ronen I. Brafman. 2024. Plug'n Play Task-Level Autonomy for Robotics Using POMDPs and Probabilistic Programs. *IEEE Robotics and Automation Letters* 9, 1 (Jan. 2024), 587–594. https://doi.org/10.1109/LRA.2023.3334682
[31] Zhutian Yang, Caelan Garrett, Dieter Fox, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2025. Guiding Long-Horizon Task and Motion Planning with Vision Language Models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*. 16847–16853. https://doi.org/10.1109/ICRA55743.2025.11128705
[32] Sungwook Yoon, Alan Fern, and Robert Givan. 2007. FF-Replan: A Baseline for Probabilistic Planning. In *Proceedings of the Seventeenth International Conference on International Conference on Automated Planning and Scheduling (ICAPS'07)*. AAAI Press, Providence, Rhode Island, USA, 352–359.
[33] Xiaohan Zhang, Zainab Altaweel, Yohei Hayamizu, Yan Ding, Saeid Amiri, Hao Yang, Andy Kaminski, Chad Esselink, and Shiqi Zhang. 2024. DKPROMPT: Domain Knowledge Prompting Vision-Language Models for Open-World Planning. https://doi.org/10.48550/arXiv.2406.17659 arXiv:2406.17659 [cs]

A PROOF OF ??

Definition 5 (Bit diffset). *Let $s, s' \in 2^F$. The* bit diffset $D(s, s')$ *is defined as the set of fluents that differ between s and s', i.e.,*

$$D(s, s') = \{f \in F | s(f) \neq s'(f)\}$$

Definition 6 (Bit-flip operator). *Let $s \in 2^F$ and $f \in F$. The* bit-flip operator *$Flip(s, f)$ returns a new state $s' \in 2^F$ such that:*

$$\forall f' \in F \quad s'(f') = \begin{cases} 1 - s(f) & \text{if } f' = f \\ s(f') & \text{otherwise} \end{cases}$$

Definition 7 (Canonical path). *Let $s, s' \in 2^F$. Let $D(s, s') = \{f_1, ..., f_n\}$ where the fluents are numbered according to a fixed total ordering over F. The* canonical path *from s to s' is the path of states $s = s_0 \rightarrow s_1 \rightarrow ... \rightarrow s_n = s'$ achieved by flipping the bits of state s_0 one by one in the order of the fluents. That is, for $i \in \{1, ..., n\}$, define $s_i = Flip(s_{i-1}, f_i)$, from $s_0 = s$ leading to $s_n = s'$.*

Lemma 1. *In Algorithm 1, let S^{ext} denote the set of states that have been extracted from the heap Q thus far. Then at the moment of max-extraction from Q, it holds that $Q \cup S^{ext} = Visited$.*

Proof. Note that every insertion to the heap is accompanied by an insertion of the same state to the visited set. All states that have ever been added to the heap are either still in the heap or have been extracted from it. Since no state is ever removed from the visited set, the lemma holds. □

Lemma 2. *In Algorithm 1, the probabilities of the states on the canonical path from s_0 to any state s are monotonically non-increasing in the states' unconstrained belief values, i.e.:*

$$\tilde{\beta}(s_0) \geq ... \geq \tilde{\beta}(s_n) = \tilde{\beta}(s)$$

Proof. Assume by contradiction that there exists $0 < j \leq n$ such that $0 \leq \tilde{\beta}(s_{j-1}) < \tilde{\beta}(s_j)$. Then all of the components that comprise the product in $\tilde{\beta}(s_j)$ are non-zero. Thus, we can safely divide by

$$\prod_{f \in F \setminus \{f_j\}} \beta_F(f)^{s_{j-1}(f)} (1 - \beta_F(f))^{1 - s_{j-1}(f)}$$

From here, we have:

$$\begin{aligned}
&\tilde{\beta}(s_{j-1}) < \tilde{\beta}(s_j) \\
\overset{1}{\iff} &\prod_{f \in F} \beta_F(f)^{s_{j-1}(f)} (1 - \beta_F(f))^{1 - s_{j-1}(f)} < \prod_{f \in F} \beta_F(f)^{s_j(f)} (1 - \beta_F(f))^{1 - s_j(f)} \\
\overset{2}{\iff} &\beta_F(f_j)^{s_{j-1}(f_j)} (1 - \beta_F(f_j))^{1 - s_{j-1}(f_j)} < \beta_F(f_j)^{s_j(f_j)} (1 - \beta_F(f_j))^{1 - s_j(f_j)} \\
\overset{3}{\iff} &\beta_F(f_j)^{s_{j-1}(f_j)} (1 - \beta_F(f_j))^{1 - s_{j-1}(f_j)} < \beta_F(f_j)^{1 - s_{j-1}(f_j)} (1 - \beta_F(f_j))^{s_{j-1}(f_j)}
\end{aligned}$$

Transition 1 follows the definition of $\tilde{\beta}$ in Eq. (1). Transition 2 is a division by all components in the product except for the one corresponding to f_j, which are non-zero, which are equal since $s_{j-1}(f) = s_j(f)$ for all $f \neq f_j$ by the definition of the canonical path in Definition 7. Transition 3 follows from the definition of the bit-flip operation in Definition 6, which states that $s_j(f_j) = 1 - s_{j-1}(f_j)$.

If $s_{j-1}(f_j) = 1$, then the above simplifies to

$$\begin{aligned}
&\beta_F(f_j)^1 (1 - \beta_F(f_j))^0 < \beta_F(f_j)^0 (1 - \beta_F(f_j))^1 \\
&\Rightarrow \beta_F(f_j) < 0.5
\end{aligned}$$

If $s_{j-1}(f_j) = 0$, then the above simplifies to

$$\begin{aligned}
&\beta_F(f_j)^0 (1 - \beta_F(f_j))^1 < \beta_F(f_j)^1 (1 - \beta_F(f_j))^0 \\
&\Rightarrow \beta_F(f_j) > 0.5
\end{aligned}$$

Since $s_{j-1}(f_j) = s_0(f_j)$, then both cases contradict the definition of s_0, which for all fluents is defined as $s_0(f) = \mathbb{1}[\beta_F(f) > 0.5]$. □

Lemma 3. *In Algorithm 1, when a state $s \in 2^F$ is extracted from the max-heap Q, then for all $s' \in 2^F$ not yet extracted from Q, it holds that $\tilde{\beta}(s) \geq \tilde{\beta}(s')$.*

Proof. Let $S_k^{\text{ext}} = \{s_0, ..., s_{k-1}\}$ be the set of the first k states extracted from Q, numbered by the order in which they were extracted. We must show that for all k and for all states $s \in S_k^{\text{ext}}$, and

for all $s' \notin S_k^{\text{ext}}$, it holds that $\beta(s) \geq \beta(s')$. We do this by induction on k.

Basis: When $k = 1$, this is the first iteration where the only value in the heap is s_0. By definition, for any $s \neq s_0$, it holds that $\tilde{\beta}(s_0) \geq \tilde{\beta}(s)$

Assumption: Assume that for some k it holds that for all $s \in S_k^{\text{ext}}$ and $s' \notin S_k^{\text{ext}}$ we have $\tilde{\beta}(s) \geq \tilde{\beta}(s')$.

Induction Step: Let $S_{k+1}^{\text{ext}} = \{s_0, ..., s_k\}$. By the induction assumption, $S_k^{\text{ext}} = S_{k+1}^{\text{ext}} \setminus \{s_k\}$ contains the top k most probable states, i.e., for all $s \in S_k^{\text{ext}}$ and for all $s' \notin S_k^{\text{ext}}$, it holds that $\tilde{\beta}(s) \geq \tilde{\beta}(s')$. It is left to show that for all $s \notin S_{k+1}^{\text{ext}}$, it holds that $\tilde{\beta}(s_k) \geq \tilde{\beta}(s)$.

Assume by contradiction that there exists a state $s \notin S_{k+1}^{\text{ext}}$ such that $\tilde{\beta}(s) > \tilde{\beta}(s_k)$. Then $s \notin Q$ because otherwise it would have been extracted before s_k. By Lemma 1, s has not yet been visited.

On the canonical path from s_0 to $s_n = s$, since s_0 has been visited and $s_n = s$ has not yet been visited, there exists $0 < j \leq n$ such that s_j has not yet been visited and s_{j-1} has.

- Since s_{j-1} is in the visited set, then by Lemma 1, it is either in the heap or has been extracted from the heap. It cannot have been extracted from the heap because then it would have been expanded, and s_j would have been visited. Therefore, s_{j-1} is in the heap.
- Since s_k was extracted but s_{j-1} was not, it follows that $\tilde{\beta}(s_k) \geq \tilde{\beta}(s_{j-1})$.
- By Lemma 2, $\tilde{\beta}(s_{j-1}) \geq \tilde{\beta}(s_n) = \tilde{\beta}(s)$.

Putting it all together, we get:

$$\tilde{\beta}(s_k) \geq \tilde{\beta}(s_{j-1}) \geq \tilde{\beta}(s) > \tilde{\beta}(s_k)$$

which is a contradiction. Thus, for all $s \notin S_{k+1}^{\text{ext}}$, it holds that $\tilde{\beta}(s_k) \geq \tilde{\beta}(s)$. □

MLSS Theorem: In Algorithm 1, when a constraint-satisfying state s is extracted from the heap Q, then for all $s' \notin b$ it holds that $\beta(s) \geq \beta(s')$, where β is the belief computed by Eq. (2).

Proof. Let s be a constraint-satisfying state extracted from the heap at some iteration. We must show that for all $s' \notin b$, it holds that $\beta(s) \geq \beta(s')$.

If s' violates the constraints, then trivially $\beta(s') = 0 \leq \beta(s)$. Otherwise, note that b is comprised of all constraint-satisfying states that were extracted from the heap before s. Therefore, s' has not yet been extracted from the heap. By Lemma 3, we have:

$$\tilde{\beta}(s) \geq \tilde{\beta}(s') \iff \frac{\tilde{\beta}(s)}{Z} \geq \frac{\tilde{\beta}(s')}{Z} \iff \beta(s) \geq \beta(s')$$

Thus the theorem holds for all $s' \in b$. □

B PROOF OF FINITE STEP CONVERGENCE

In this section, we prove Corollary 1 via Theorem 2.

B.1 Proof of Theorem 2

Theorem Statement: Let δ be the cumulative factored belief error function. Then for all $s \in 2^F$ and $\beta_F : 2^F \to [0, 1]^F$, if $\delta(\beta_F, s) < \theta$, then s is in the MLSS.

Proof. Let $s \in 2^F$ and let $\beta_F : 2^F \to [0, 1]^F$. For simplicity, we denote $\delta = \delta(\beta_F, s)$.

Let $\varepsilon_f = |\beta_f - s(f)|$ be a single fluent belief error, and let $W(s) = \prod_{f \in F} \beta_f^{s(f)} (1 - \beta_f)^{1-s(f)}$ be the unnormalized probability weight of state s.

If $s(f) = 1$ then

$$\varepsilon_f = |\beta_f - s(f)| = 1 - \beta_f$$
$$\Rightarrow \beta_f = 1 - \varepsilon_f$$

and if $s(f) = 0$ then

$$\varepsilon_f = |\beta_f - s(f)| = \beta_f$$
$$\Rightarrow 1 - \beta_f = 1 - \varepsilon_f$$

Then by the Weierstrass product inequality, the unnormalized weight assigned to s is:

$$W(s) = \prod_{f \in F} (1 - \varepsilon_f) \geq 1 - \sum_{f \in F} \varepsilon_f = 1 - \delta$$

β is normalized by normalizing factor $0 < Z \leq 1$. Thus:

$$\beta(s) = \frac{W(s)}{Z} \geq W(s) \geq 1 - \delta$$

Assume by contradiction that s is not in the MLSS. Then:

$$\beta(s) \leq \sum_{s \notin MLSS} \beta(s) \leq 1 - \theta$$

From the previous inequality:

$$1 - \theta \geq \beta(s) \geq 1 - \delta$$
$$\Rightarrow \delta \geq \theta$$

This contradicts the theorem's assumption that $\delta < \theta$, and thus s must be in the MLSS.

□

B.2 Proof of Corollary 1

Theorem Statement: Let $\rho > 0$ be the minimum visibility rate. If the VLM is weakly calibrated with error margin γ, then there exists T such that a conformant plan for the MLSS is satisficing from the true current state.

Proof. Let $l_{f,t}$ be the log odds of the correct assignment of f at time t of the evidence provided by the VLM. Since the VLM is weakly calibrated, then:

$$p_{\text{correct},t} - (1 - p_{\text{correct}}, t) \geq p_{\text{correct},t} - p_{\text{incorrect},t} \geq \gamma$$
$$\Rightarrow p_{\text{correct},t} \geq \frac{\gamma}{2} + 0.5$$

This implies that $1 - p_{\text{correct},t} \leq 0.5 - \frac{\gamma}{2}$. Thus:

$$\frac{p_{\text{correct},t}}{1 - p_{\text{correct},t}} \geq \frac{0.5 + \frac{\gamma}{2}}{0.5 - \frac{\gamma}{2}}$$
$$\Rightarrow \log\left(\frac{p_{\text{correct},t}}{1 - p_{\text{correct},t}}\right) \geq \log\left(\frac{0.5 + \frac{\gamma}{2}}{0.5 - \frac{\gamma}{2}}\right)$$
$$\Rightarrow l_{f,t} \geq \log\left(\frac{0.5 + \frac{\gamma}{2}}{0.5 - \frac{\gamma}{2}}\right) > 0$$

Then L is bounded by a strictly positive constant. Denote $C = \log\left(\frac{0.5+\frac{\gamma}{2}}{0.5-\frac{\gamma}{2}}\right)$.

Let $L_{f,t}$ be the log odds of the correct assignment of f at time t according to factored belief $\beta_{F,t}$. The logarithmic pooling update rule is $L_{f,t} = L_{f,t-1} + l_{f,t}$. Assume W.L.O.G. that the initial belief $\beta_{F,0}$ is uniform (because in any case, it is constant). Since we only update fluents that are visible, the log odds value at time T is:

$$\begin{aligned} L_{f,T} &= \log\left(\frac{\beta_{f,0}}{1-\beta_{f,0}}\right) + \sum_{t=1}^{T} \mathbb{1}_f(t) \cdot l_{f,t} \\ &= \sum_{t=1}^{T} \mathbb{1}_f(t) \cdot l_{f,t} \end{aligned}$$

where $\mathbb{1}_f(t)$ is an indicator function for fluent f being visible at timestep t. By the definition of ρ, all fluents appear at least ρT times within T timesteps. Thus:

$$L_{f,T} \geq \rho T \cdot C$$

Let δ_t be the cumulative factored belief error at time t, and let $\varepsilon_f = |\beta_f - s^*(f)| = 1 - p_{\text{correct}}$ where s^* is the true current state. Then:

$$\begin{aligned} L_{f,t} &= \log\left(\frac{p_{\text{correct}}}{1-p_{\text{correct}}}\right) \\ L_{f,t} &= \log\left(\frac{1-\varepsilon_{f,t}}{\varepsilon_{f,t}}\right) \\ e^{L_{f,t}} &= \frac{1-\varepsilon_{f,t}}{\varepsilon_{f,t}} \\ \varepsilon_{f,t} &= \frac{1}{1+e^{L_{f,t}}} \leq e^{-L_{f,t}} \leq e^{-\rho T \cdot C} \end{aligned}$$

We want the cumulative factored belief error to be less than θ. Note that:

$$\delta_t = \sum_{f \in F} \varepsilon_{f,t} \leq \sum_{f \in F} e^{-\rho T \cdot C} = |F| \cdot e^{-\rho T \cdot C}$$

Then it is enough to find T such that

$$\begin{aligned} &|F| \cdot e^{-\rho T \cdot C} < \theta \\ \Leftrightarrow &-\rho T \cdot C < \log\left(\frac{\theta}{|F|}\right) \\ \Leftrightarrow &T > \frac{\log(|F|) - \log(\theta)}{\rho \cdot C} \end{aligned}$$

Since all values $|F|, \theta, C$, and ρ are all constant, there exists a finite T such that this inequality always holds. Therefore, after the Tth belief update, s^* is in the MLSS, and so a conformant plan for all states in the MLSS is also a satisficing plan from s^*. □

C PROOF OF ??

Theorem statement: Let $\text{unsafe}(\pi)$ denote the event that policy π executes an unsafe action. Then,

$$\Pr(\text{unsafe}(\pi_{det})) \geq \Pr(\text{unsafe}(\pi_{cpp})) + (\theta - \beta(s_{max}))$$

where s_{max} is a maximizer of β.

Proof. For belief β, π_{det} generates a plan that is verified to be valid for the most likely state, denoted s_{max}. Thus,

$$\Pr(\text{unsafe}(\pi_det)) = 1 - \beta(s_{max})$$

Policy π_{cpp} generates a plan that is verified for all states in the MLSS, denoted β_θ. By the definition of β_θ:

$$\begin{aligned} &\sum_{s \in \beta_\theta} \beta(s) \geq \theta \\ \Rightarrow &\Pr(\text{unsafe}(\pi_{cpp})) = 1 - \sum_{s \in \beta_\theta} \beta(s) \leq 1 - \theta \end{aligned}$$

From here:

$$\begin{aligned} &\Pr(\text{unsafe}(\Pr det)) \\ =&1 - \beta(s_{max}) \\ =&(1-\theta) + (\theta - \beta(s_{max})) \\ \geq&\Pr(\text{unsafe}(\pi_{cpp})) + (\theta - \beta(s_{max})) \\ =&\Pr(\text{unsafe}(\pi_{cpp})) + (\theta - \max_{s \in 2^F} \beta(s)) \end{aligned}$$

□

D ACRONYMS

CP conformant planning. 3, 4, 6
CPP conformant probabilistic planning. 2–6
CWA Closed World Assumption. 1

MLSS Most Likely Subset of States. 4–6, 10, 11

PDDL Planning Domain Definition Language. 4, 7

RoVLaP Robust Vision-Language Planning. 2, 3, 6–8
RVP Robust Visual Planning. 2, 3, 6, 7

S3E Semantic Symbolic State Estimation. 7
STRIPS Stanford Research Institute Problem Solver. 3

ViLa Vision Language. 7, 8
VLA Vision-Language-Action. 2, 7
VLM Vision-Language Model. 1–8, 10