When to Act, Ask, or Learn: Uncertainty-Aware Policy Steering

Anonymous Authors

***Abstract*—Policy steering is an emerging way to adapt robot behaviors at deployment-time: a learned verifier analyzes low-level action samples proposed by a pre-trained policy (e.g., diffusion policy) and selects only those aligned with the task. While Vision-Language Models (VLMs) are promising general-purpose verifiers due to their reasoning capabilities, existing frameworks often assume these models are well-calibrated. In practice, the overconfident judgment from VLM can degrade the steering performance under both high-level semantic uncertainty in task specifications and low-level action uncertainty or incapability of the pre-trained policy. We propose *uncertainty-aware policy steering* (UPS), a framework that jointly reasons about semantic task uncertainty and low-level action feasibility, and selects an uncertainty resolution strategy: execute a high-confidence action, clarify task ambiguity via natural language queries, or ask for action interventions to correct the low-level policy when it is deemed incapable at the task. We leverage conformal prediction to calibrate the composition of the VLM and the pre-trained base policy, providing statistical assurances that the verifier selects the correct strategy. After collecting interventions during deployment, we employ residual learning to improve the capability of the pre-trained policy, enabling the system to learn continually but with minimal expensive human feedback. We demonstrate our framework through experiments in simulation and on hardware, showing that UPS can disentangle confident, ambiguous, and incapable scenarios and minimizes expensive user interventions compared to uncalibrated baselines and prior human- or robot-gated continual learning approaches.**

I. Introduction

Imitation-based generative robot policies, such as diffusion policies and vision-language-action (VLA) models, are a promising way to model low-level action distributions conditioned on perceptual (and language) inputs. However, directly executing samples from the learned action distribution does not always lead to success at deployment time. This has motivated the paradigm of *policy steering* (also called test-time compute) which adapts the behavior of a pre-trained policy at deployment time through sampling and verification.

Recent works [30, 12] demonstrated that vision-language models (VLMs) can be harnessed as "open world" verifiers during policy steering, reasoning about the outcomes of action samples and selecting those which align most with task instructions. However, to-date, all approaches have implicitly assumed that the verifier is well-calibrated: i.e., it selects an appropriate action sample (or correctly rejects all candidates) with high confidence. In practice, this assumption often fails. VLM verifiers can be overconfident when task instructions are ambiguous or under-specified, leading to confident selection of misaligned behaviors. Moreover, when the low-level policy is fundamentally *incapable* of accomplishing the task under the current conditions, all candidate samples may be unsatisfactory. Our experiments show that an uncalibrated verifier may still select one of these faulty options rather than identifying the limitation and stopping execution.

In this work, we propose ***U**ncertainty-aware **P**olicy **S**teering* (**UPS**), a method that maps a VLM verifier's uncertainty over action samples into a resolution strategy: **execute** a high-confidence action sample, **clarify** task ambiguity through natural-language interactions, or request to **re-train** the low-level policy when it is incapable. Specifically, we calibrate the composition of the VLM verifier and the pre-trained policy using conformal prediction [2], returning a *set* of action sample(s) and possibly a "none of the above" option. This set is constructed such that, with a user-specified level $1-\epsilon$, the verifier can steer the low-level policy to success in $1-\epsilon$ of scenarios by either executing the right action sample or selecting the appropriate resolution strategy (asking clarification questions when ambiguous or eliciting human interventions when incapable). Our conformal prediction approach also minimizes the average size of prediction sets, ensuring that the robot selectively chooses to ask semantic clarification questions (when the set size is > 1) or chooses to ask for low-level retraining (when the set is a singleton with "none of the above"). When low-level human intervention is required, we improve the policy via residual learning [10, 31], enabling continual improvement of the robot's capabilities while mitigating catastrophic forgetting.

We evaluate **UPS** in a series of simulation and hardware experiments with robotic manipulation. Compared to prior policy steering methods that do *not* calibrate the verifier, we show 30% improvement in ambiguous scenarios. On the uncertainty-quantification side, we show that our approach to conformal prediction, including a new score function with VLMs, yields strong empirical coverage guarantees with tight prediction sets compared to prior UQ methods for VLMs. On the continual learning side, we show that **UPS**'s ability to reason about both high-level semantic uncertainty and low-level action uncertainty enables it to more effectively balance cheap language queries vs. expensive low-level action interventions from a human. Furthermore, compared to human-[11] or robot-gated [19] interactive imitation learning approaches, the data collected by **UPS** enables the robot to improve over successive deployment rounds while minimizing human effort.

II. Related Work

Policy Steering in Robotics. Recent work increasingly use VLMs as deployment-time verifiers to steer robot policies by

scoring multiple sampled trajectories [30, 12, 29]. However, these pipelines suffer from the same problems as LLM-based verification, including poor calibration and systematic agreement bias, where plausible but incorrect actions are blindly accepted [15, 1]. These failures are especially problematic in robotics because verifier outputs are often reduced to a scalar score or accept/reject decision, obscuring *what* is uncertain and *how* the system should respond. Our work addresses this gap by modeling semantic uncertainty in VLM verification and mapping different uncertainty modes to distinct resolution strategies, from high-level clarification to low-level data collection.

Uncertainty Quantification for Learned Robot Policies. While uncertainty quantification (UQ) has been a long-standing problem in robotics and machine learning [27, 7], in this paper, we focus on learning-based robot policies that generate low-level actions from multimodal inputs [6, 21] such as images and language instructions. Prior work typically addresses UQ at two decoupled levels of abstraction: semantic ambiguity and low-level action uncertainty. For high-level semantic uncertainty, approaches like KnowNo [21] and RCIP [13] employ conformal prediction to provide statistical guarantees that the generated plans (or intent predictions) capture the user's true instruction. However, these methods largely operate in discrete textual or symbolic space with an unrealistic assumption that the low-level policy can always execute the chosen plan. A separate body of work quantifies uncertainty directly within the continuous action space of learned policies, such as VLA or diffusion models [26, 34]. While these methods can flag distribution shifts or execution noise [32], low-level action uncertainty does not always correlate with task-level failures (e.g., there are multiple ways to successfully grasp a cup). Our work takes a step towards more tightly coupling the uncertainty of the low-level action policy (approximated by repeatedly sampling from the policy) with high-level semantic uncertainty about the task (quantified by how well the action outcomes align with the user's instruction).

Continual Learning via Interactive Imitation. Interactive Imitation Learning (IIL) improves policies through human feedback. The DAgger family of methods [23, 9] typically relies on the human operator to know when to intervene, or on action-level uncertainty signals to trigger interventions, which has challenges as described in the above section. In contrast, our approach leverages calibration to capture semantic uncertainty over different meaningful phases (e.g., grasping, placing) to minimize human feedback. Recent work also uses the imagination of a world model to evaluate long-horizon execution failures [14]. However, these methods typically evaluate a single policy trajectory. Because they do not sample multiple rollouts within the world model, they can't effectively "search" through the base policy's distribution to find an alternative action plan; this can result in asking for human interventions even when a feasible solution exists within the support of the policy distribution. Our method mitigates this by performing action sampling and calibrated verification. Finally, to adapt efficiently from human intervention, recent works utilize residual policy learning [10, 31] instead of fine-tuning the entire base model. We adopt this paradigm as it enables our system to integrate user corrections into the generation-imagination loop without destroying the diverse capabilities of the base policy.

III. Problem Formulation

Setup & Notation. We define the user's task instruction $\mathcal{L}$ as a single sentence defining the goal of a long-horizon task that involves multiple T-timestep subtasks. The robot's observation space $o \in \mathcal{O} := \mathcal{I} \times \mathcal{S}$ integrates RGB images I with proprioceptive states s, such as end-effector poses. Control is governed by a pre-trained base policy $\pi(a \mid o_t)$, typically a generative visuomotor model such as a diffusion policy [6] or a vision-language-action model [5]. At any timestep t, the policy generates a short action chunk $a_{t:t+H}$ ($H \ll T$), which is aggregated into a full sequence $\mathbf{a}_t = a_{t:t+T}$ with multiple continuous generations. Given an observation o_t, sampling from the policy $\mathbf{a}_t \sim \pi(\cdot \mid o_t)$ and executing the plan yields a corresponding future observation sequence $\mathbf{o}_t \sim \mathbb{P}(\cdot \mid o_t, \mathbf{a}_t)$. This formulation allows us to evaluate the policy's predicted outcomes against the high-level task instruction across multiple stages of execution.

Problem. Given the robot's current observation o and K *i.i.d.* samples from the base policy, $\{\mathbf{a}^i\}_{k=1}^K \sim \pi(\mathbf{a} \mid o)$, our *uncertainty-aware policy steering* problem seeks to return the index $y \in \mathcal{Y} = \{1, ..., K, K+1\}$ of the action sample that best matches the textual task description $\mathcal{L}$ *or* return a $K+1$-th index which indicates "none of the above" (i.e., no samples accomplish the instructed task). More formally, we seek to solve the following optimization problem:

$$y^\star = \arg\max_{y \in \mathcal{Y}} \mathbb{P}(y \mid \{\mathbf{a}^k\}_{k=1}^K, o; \mathcal{L}). \tag{1}$$

In general, the distribution $\mathbb{P}(y \mid \{\mathbf{a}^k\}_{k=1}^K, o; \mathcal{L})$ above is very hard to characterize, which is why prior works [21, 30] have turned to vision-language models (VLMs) due to their ability to simultaneously reason about both natural language instructions and visual observations. However, prior work [30] shows that effective zero-shot use of VLMs to approximate $\mathbb{P}(y \mid \{\mathbf{a}^k\}_{k=1}^K, o; \mathcal{L})$ is not trivial, since the model must implicitly translate low-level actions into outcomes (i.e., predict $\{\mathbf{a}^k\}_{k=1}^K \to \{\mathbf{o}^k\}_{k=1}^K$), reason about the outcomes (i.e., alignment between the outcomes $\{\mathbf{o}^i\}_{k=1}^K$ and the text of the task $\mathcal{L}$), as well as select the correct index.

Following prior work [30, 29], we factorize the problem into two stages: first, predict the outcome of each low-level action and translate the outcomes into textual *outcome narrations*; given these narrations, we ask a VLM to solve a multiple-choice Q&A problem and select the action whose outcome narration best aligns with the task. Mathematically, our problem becomes:

$$y^\star = \arg\max_{y \in \mathcal{Y}} \mathbb{E}_{\{\ell^k\}_{k=1}^K \sim \mathbb{P}(\cdot \mid \{\mathbf{a}^k\}_{k=1}^K, o)} \Big[\mathbb{P}(y \mid \{\ell^k\}_{k=1}^K; \mathcal{L}) \Big]. \tag{2}$$

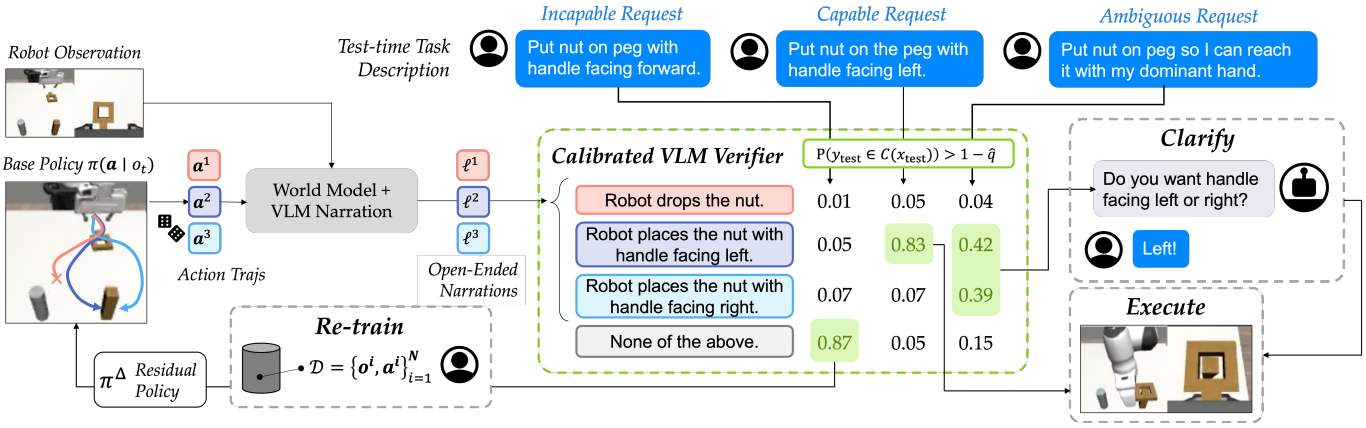


Fig. 1: **Uncertainty-Aware Policy Steering.** Our framework calibrates the VLM verifier used for policy steering via conformal prediction. This enables the VLM to select an appropriate way to resolve uncertainty, from querying the end-user in natural language to asking to re-train the low-level control policy.

Here, each ℓ^k is a language description of the outcome induced by the low-level action $\mathbf{a}^k$ executed from an initial observation. For example, ℓ^k = "The robot places the nut with the handle facing left." as shown in Fig. 1. In practice, these narrations are obtained by first *predicting future observations* $\mathbf{o}_t \sim \mathbb{P}(\cdot|o_t, \mathbf{a}_t)$ (e.g., via a world model, shown in Sec. IV), and then the predicted observations are described in text via video captioning [25]. By lifting the outcomes of low-level actions into textual descriptions, the VLM's reasoning capabilities can be used more effectively to verify which index $y \in \mathcal{Y}$ to choose; in other words, the VLM can be used as a proposal distribution in the optimization objective: $\mathbb{P}(y \mid \{\ell^k\}_{k=1}^K; \mathcal{L})$.

Challenges & Opportunities. A central challenge with Eq. 2 is the assumption of a well-calibrated VLM verifier. Even with perfect outcome narrations to choose from, VLMs are often overconfident evaluators [3, 18] which can undermine the quality of the policy steering. Moreover, the task descriptions $\mathcal{L}$ from the user may be inherently *ambiguous* or *underspecified*; a VLM verifier which confidently selects an incorrect behavior narration can lead to misaligned or even unsafe actions being executed. A second challenge is that when the policy is *incapable*, all K action samples from the policy in Eq. 2 will have unsatisfactory outcomes for the task instruction $\mathcal{L}$; an uncalibrated VLM verifier may choose one of the faulty samples instead of rejecting them all.

Finally, even if uncertainty or incapability is detected, the robot should *not* always stop operation and instead should generate an appropriate resolution strategy. However, actionable strategies for resolving uncertainty can range from simple clarifications with the user (e.g., "Did you want the nut handle facing left or right?") to far more demanding interventions such as re-training the low-level control policy (e.g., refining the precision of the skill to place a nut on a peg, or learning a new skill for placing with a novel orientation). In other words, the source of uncertainty can lie at different levels of abstraction, from *semantic ambiguity* (e.g., multiple indices seem plausible for the task) to the base policy's *incapability*. Our unified framework seeks to both calibrate the VLM verifier and equip the robot with appropriate resolution strategies.

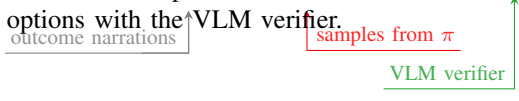
Aside: Comparison to KnowNo [21]. A closely related work [21] similarly performs uncertainty quantification on an LLM task planner that reasons about textual plans and asks questions when highly uncertain. However, it assumes an idealized low-level policy that executes any chosen plan accurately. We relax this assumption by grounding the uncertainty quantification in the capabilities of low-level policy. We achieve this because our textual plans (that the VLM verifier reasons about) directly correspond to outcomes induced by samples from low-level policy in Eq. 2, as well as an explicit "incapability" option. Thus, our uncertainty quantification jointly reasons about high-level semantic intent uncertainty and low-level action uncertainty. Together, our system not only resolves what the user wants, but also identifies whether the policy can reliably accomplish the task.

IV. Approach: Uncertainty-Aware Policy Steering

Our uncertainty-aware policy steering approach calibrates the VLM verifier to select, with high user-specified confidence, one of three uncertainty resolution strategies: **execute** a high-confidence action that fulfills the task; **clarify** task ambiguity when multiple actions seem plausible; or **re-train** the low-level policy via interactive imitation learning when it is incapable. Our overall framework is visualized in Fig. 1, and we describe each component in the following subsections.

A. VLM-in-the-loop Policy Steering

We follow the framework from [30, 29] and pose VLM-in-the-loop steering as an open-ended multiple-choice Q&A problem. This involves three stages: predicting the outcomes $\mathbf{o}$ of an action sample $\mathbf{a}$ with a world model, generating behavior narrations ℓ for the predicted outcomes, and selecting from these options with the VLM verifier.



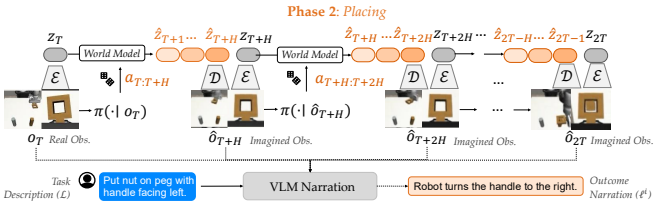


Fig. 2: **Outcome Prediction & Narration.** The policy and the world model are interleaved to predict long-horizon outcomes induced by the low-level policy. Decoded observations are fed into a VLM which narrates the outcomes in text.

Outcome Prediction. We leverage latent world models [8, 33] to predict the outcomes of low-level action samples directly from a high-dimensional observation, o_t. The world model $\mathcal{W}_\phi$ consists of an encoder $\mathcal{E}_\phi : \mathcal{O} \to \mathcal{Z}$ which encodes an observation o_t, a latent dynamics model $f_\phi : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$ which predicts latent state transitions, and a decoder $\mathcal{D}_\phi : \mathcal{Z} \to \mathcal{O}$ which reconstructs an observation from latent space. Recall that most low-level policies [6, 5] predict relatively short-horizon action "chunks". This presents a challenge for policy steering, since on very short time-horizons, the outcomes are often indistinguishable for the verifier.

To address this, we interleave action generation and imagination to obtain a longer-horizon action-outcome sequence that is informative for the verifier to reason about (visualized in Figure 2). Specifically, given the current *real* observation o_t, we query $a_{t:t+H} \sim \pi(\cdot \mid o_t)$ to generate an action chunk of length H and pass this to the world model to obtain predicted future observations: $\hat{o}_{t+1:t+H} = \mathcal{W}_\phi(o_t, a_{t+1:t+H})$. The last predicted observation $\hat{o}_{t+H}$ is provided as input to the policy $\pi(\cdot \mid \hat{o}_{t+H})$ to generate the next action chunk. We repeat this process T/H times and aggregate all action sequences together into one T-length sequence $a_{t:t+T}$ with corresponding imagined observations $\hat{o}_{t:t+T}$. By executing this process in parallel for K samples, we produce a set of predicted observation sequences $\{\hat{o}^k_{t:t+T}\}_{k=1}^K$ that are (approximate) future outcomes.

Narration. Next, the imagined outcomes $\{\hat{o}^k_{t:t+T}\}_{k=1}^K$ are translated into textual narrations by leveraging the strong reasoning capabilities of the VLM verifier. In our experiments, we utilize a Gemini model [25] to produce these narrations. Let $\mathcal{T}^{\text{VLM}}$ be the VLM narration model and for any $\hat{\mathbf{o}}^k_t \in \{\hat{\mathbf{o}}^k_t\}_{k=1}^K$, let the corresponding narration be $\ell^k_t = \mathcal{T}^{\text{VLM}}(\hat{\mathbf{o}}^k_t, \mathcal{L})$. We augment the set of narrations with a $(K+1)$-th option stating that "none of the above samples are correct" to ensure the system can detect incapability.

Verification. Finally, the VLM verifier selects an option from the set of narrations (along with the "none" option) by answering an open-ended multiple-choice problem over these narrations. Mathematically, this amounts to the VLM selecting the most likely single token corresponding to one of the multiple choice options: $y_t = \arg\max_{y \in \mathcal{Y}} p^{\text{VLM}}(y \mid \{\ell^k_t\}_{k=1}^{K+1}, \mathcal{L})$. This formulation aligns with the VLM's native log-likelihood loss and, as noted in [21], and eliminates length bias when estimating the probability of open-ended textual generations.

B. VLM Verifier Uncertainty Quantification

We employ conformal prediction (CP) [2] to quantify the VLM verifier's uncertainty during policy steering. Specifically, our goal is to use the potentially unreliable model likelihoods in p^{VLM} to construct tight prediction sets which are provably guaranteed to contain the correct action with probability of at least $1-\epsilon$, where ϵ is a user-defined error budget.

Conformal Prediction. Given some *input* x to a prediction model, conformal prediction aims to construct a prediction set $C(x) \subseteq \mathcal{Y}$ on the *outputs* of that model that contains the true label y with a probability of at least $1-\epsilon$ while minimizing the set size $|C(x)|$. This calibration requires a dataset $(x, y) \in \mathcal{D}_{\text{calib}}$ of independent and identically distributed ($i.i.d.$) input-output pairs drawn from the same distribution as the test domain. However, because we are in the sequential-decision-making setting, any input to the VLM verifier depends on prior steering decisions and thus are not $i.i.d.$.

To maintain formal coverage guarantees despite these temporal dependencies, we perform sequence-level calibration. Assume that the VLM verifier is called M times during the task. This means we have M true observations that the verifier uses, M sets of behavior narrations, and—for calibration—M true labels about which behavior narration(s) are correct at each timestep. Let the sequence-level calibration dataset be structured as follows:

$$\{o_i, \{\{\ell^k\}_{k=1}^K\}_i, \mathcal{L}, \mathcal{Y}_i^*\}_{i=0}^M \in \mathcal{D}_{\text{calib}}, \quad |\mathcal{D}_{\text{calib}}| = N.$$

For notational simplicity, denote any single *input* $x_i = (o_i, \{\{\ell^k\}_{k=1}^K\}_i, \mathcal{L})$ and (set of) ground-truth label(s) $\mathcal{Y}_i^*$ obtained from an end-user. Then, denote a *sequence* of inputs and labels as $\bar{x} = (x_0, \ldots, x_M)$ and $\bar{y} = (\mathcal{Y}_0^*, \ldots, \mathcal{Y}_M^*)$. To construct $\mathcal{D}_{\text{calib}}$, we sample multiple initial observations and different task instructions, use our interleaved generation-imagination loop to obtain K candidate narrations and the "none" option, annotate the correct set of indices, and execute the correct action; this results in the next observation that the verifier sees and serves as the beginning of next generation-imagination loop, and so on.

Given this calibration dataset $\mathcal{D}_{\text{calib}}$, we use the VLM verifier's learned distribution p^{VLM} to compute the *non-conformity score*, which is defined for any data point $(\bar{x}, \bar{y}) \in \mathcal{D}_{\text{calib}}$ as:

$$\kappa(\bar{x}, \bar{y}) = 1 - \min_{x_i \in \bar{x}} \min_{y \in \mathcal{Y}_i^*} p^{\text{VLM}}(y \mid x_i). \tag{3}$$

We compute this quantity for all N sequences in the calibration dataset. There are a few points worth noting here. First, intuitively, for any calibration data point, the higher the score, the less the data "conforms" to the training data of the verifier. Second, we design this score function to compute a *minimum* over the entire sequence (outer min in Eq. 7) to ensure coverage guarantees despite these temporal dependencies. Finally, the inner minimium in Eq. 7 is a conservative score designed to penalize the model for not including *all* suitable options in the prediction set. From a user perspective, this avoids

scenarios where the verifier guesses the user's unstated intent and incentivizes the verifier to ask the user about the entire set of plausible choices.

Finally, CP calibrates the VLM verifier by selecting the $\hat{q} = \frac{\lceil (N+1)(1-\epsilon) \rceil}{N}$ empirical quantile of the nonconformity scores $\{\kappa^1, ..., \kappa^N\}$ and uses $\hat{q}$ to construct the prediction set $C(x)$. This includes all labels that the VLM verifier is at least $1-\hat{q}$ confident about, ensuring the $1-\epsilon$ coverage guarantee [27, 21].

Deployment Time. At deployment time, the robot is given a new task description $\mathcal{L}^{\text{test}}$. Given the current observation o_i^{test}, we obtain $x_i^{\text{test}} = (o_i^{\text{test}}, \{\{\ell^k\}_{k=1}^K\}_i, \mathcal{L}^{\text{test}})$ via the same action sampling, prediction, and narration pipeline described above. We query the VLM verifier to estimate the probability $p^{\text{VLM}}(y \mid x_i^{\text{test}})$ for each choice $y \in \mathcal{Y} = \{1, ..., K, K+1\}$, and the prediction set is generated via

$$C(x_i^{\text{test}}) = \{y \in \mathcal{Y} \mid 1 - p^{\text{VLM}}(y \mid x_i^{\text{test}}) \leq \hat{q}\} \tag{4}$$

i.e., selecting any index whose probability is larger than the calibrated empirical quantile $1-\hat{q}$. Assuming the narrations ℓ faithfully represent the outcomes of actions, we formally state the statistical assurance for our calibration procedure.

Theorem 1 (Verification Coverage Guarantee): Let $D_{\text{calib}} = \{\{(x_i^n, \mathcal{Y}_i^n)\}_{i=0}^M\}_{n=1}^N$ be the calibration dataset with non-conformity score:

$$\kappa^n := \kappa(\bar{x}^n, \bar{y}^n) = 1 - \min_{x_i \in \bar{x}^n} \min_{y \in \mathcal{Y}_i^n} p^{\text{VLM}}(y \mid x_i^n)$$

and let $\hat{q}$ be the $\frac{\lceil (N+1)(1-\varepsilon) \rceil}{N}$–quantile of $\{\kappa^n\}_{n=1}^N$. For a test point x_i^{test} at any time when the VLM verifier is called $i \in [M]$, where the correct set of labels is $\mathcal{Y}_i^* \subseteq \mathcal{Y}$, define

$$C(x_i^{\text{test}}) = \{y : 1 - p^{\text{VLM}}(y \mid x_i^{\text{test}}) \leq \hat{q}\}.$$

Then $\mathbb{P}(\mathcal{Y}_i^* \subseteq C(x_i^{\text{test}})) \geq 1 - \varepsilon$.

Proof. See Appendix A. We follow the sequence-level calibration proof from [21], but show that our non-conformity score includes all the ground-truth labels in our prediction set.

Shaping the VLM's Distribution *Before* CP. While in theory conformal prediction provides distribution-free coverage guarantees, the tightness of the resulting prediction sets depends on the non-conformity score. Our score in Eq. 7 directly depends on the VLM's distribution, p^{VLM}. Naively querying a VLM for likelihoods over multiple choices results in significantly overconfident distributions, especially in ambiguous or incapable scenarios. Although CP can compensate for this, it leads to overly-large prediction sets that are not informative and result in the robot over-asking for help. Thus, the raw softmax probabilities from the logits of the VLM verifier are *not* an ideal way to inform our score function p^{VLM} from Eq. 7.

Our key idea is shape p^{VLM} even *before* calibration by *factorizing* the VLM's reasoning process such that the uncertainty is more explicit. Our factorization is inspired by Bayesian models of human intent [4, 20] which decouple inferring human intent (e.g., unstated aspects of the task) and the likelihood of behaviors given a specific intent:

$$p^{\text{VLM}}(y \mid \{\ell^k\}_{k=1}^K; \mathcal{L}) = \sum_{\theta \in \Theta} \mathbb{P}(y \mid \{\ell^k\}_{k=1}^K, \theta) \mathbb{P}(\theta \mid \mathcal{L}). \tag{5}$$

Given the original task instruction $\mathcal{L}$ (e.g., "Put the nut on the peg with its handle facing my dominant-hand"), we query the VLM three times to estimate each component of Eq. 5. First, we ask the model to hypothesize a set of hidden human intents $\theta \in \Theta$ (e.g., "Put the nut on the peg with its handle facing Θ =[left/right/front/back]") alongside their respective probabilities, $\mathbb{P}(\theta \mid \mathcal{L})$. Then, we ask the VLM to infer the likelihood of each option, *given a hypothesized user intent*: $\mathbb{P}(y \mid \{\ell^k\}_{k=1}^K, \theta)$. The final distribution is obtained by marginalizing over the space of user intents.

Intuitively, this factorization leverages the commonsense reasoning abilities of the VLM in a structured way to re-calibrate p^{VLM} intrinsically even before non-conformity score computation. We find this to be especially useful in incapable scenarios or when one behavior mode is underrepresented in the action samples. For example, consider the scenario where all action samples place the nut with the handle facing right (i.e., ℓ^k = "The nut's handle faces right", $\forall k \in \{1, \ldots, K\}$). Naively querying p^{VLM} results in an extremely biased distribution in favor of selecting the option with handle-facing-right, ignoring the possibility that the user is left-handed. In our factorized approach, the VLM is asked to reason about hidden aspects of the task or user intents, and thus includes the possibility that the user is right *or* left-handed in Θ. This reshapes the final distribution over options, forcing the VLM to place non-trivial probability mass on the $K+1$th "none" option. In turn, this results in the reduced overconfidence of the VLM in incapable scenarios and enables the robot to ask for help only when necessary. In ambiguous scenarios, this same principle helps re-weight the distribution over several equally likely behaviors. In Sec. V, we show empirically that our factorization results in p^{VLM} that is substantially more calibrated even before conformal adjustment, and outperforms chain-of-thought reasoning (CoT) [28]. Ultimately, we obtain non-trivial prediction sets with high empirical coverage while also reducing the help rate.

C. Resolving Uncertainty: From High-level Clarifications to Low-Level Continual Learning

At deployment-time, the prediction sets constructed by our CP procedure provide the robot with a principled measure of both semantic task uncertainty and policy incapability, implying separate resolution strategies. Importantly, our statistical guarantee from Theorem 1 ensures that the VLM verifier can steer the low-level policy to success in $1-\epsilon$ proportion of scenarios by either executing the correct action sample or selecting the correct resolution strategy. We describe each strategy in detail below.

Confident and Capable Policy. If the prediction set is a singleton and contains an index corresponding to one of the low-level action plans, i.e. $|C(x_i^{\text{test}})| = 1$ and $K+1 \notin C(x_i^{\text{test}})$,

then the steering system confidently executes this sequence of actions until the next verification step (bottom right, Fig. 1).

Resolving Task Uncertainty with Textual Clarifications. If the set size $|C(x_i^{\text{test}})| > 1$, the verifier has identified high-level semantic uncertainty in the task and asks a textual clarification question. For example, consider the right part of Fig. 1. Both the handle facing left and the handle facing right options are included in the prediction set when the user gives an ambiguous task description $\mathcal{L}$ ="*put the nut on the peg so that I can reach it with my dominant hand*". After asking the user to state their preferences about options, the VLM summarizes user's answer $\tilde{\mathcal{L}}$ and replaces the original ambiguous task instruction. This verify and clarify loop repeats and continues until $|C(x_i^{\text{test}})| = 1$. The right part of Fig. 1 demonstrates this process of resolving the task uncertainty.

Resolving Policy Incapability with Continual Learning. When the CP prediction set contains only "*none of the above*", i.e., $C(x_i^{\text{test}}) = \{K+1\}$, the VLM verifier detects that none of the action samples align with the user-requested task with high probability. This triggers a low-level resolution strategy via interactive imitation learning. Specifically, the robot randomly selects a trajectory and explicitly asks human supervision to correct potential failures, as shown in Fig 6. This approach improves upon standard interactive imitation learning methods [19, 11] in two ways. First, unlike human-gated methods [11] that demand constant monitoring of every trajectory, our robot only requests help when the policy is incapable in *all* samples. Second, unlike low-level action-uncertainty methods [19], our approach gates interventions based on semantic task alignment rather than low-level action variance; this avoids scenarios where the low-level policy has noisy action predictions, but all result in good outcomes (i.e., the policy does not need to be re-trained).

We employ residual policy learning [24, 10] and train a lightweight model, $\Delta a \sim \pi^{\text{residual}}(\cdot \mid o, a)$, which predicts the difference between human corrections and base policy outputs alongside a gating classifier that determines when this correction is active. At re-deployment, we sample K actions from the frozen base policy and mix the residual policy into half of the samples (when triggered by the gating classifier), while half of the samples remain unmodified. This sampling strategy mitigates catastrophic forgetting by explicitly retaining the base policy's distribution. Finally, because the policy's action distribution shifts after residual policy learning, we recalibrate the VLM verifier, thereby closing the learning loop.

V. Simulation & Hardware Experiments

We evaluate our framework in both a benchmark simulation task and in robotic hardware with a Franka manipulator. In all our experiments, the base policy is an image-conditioned diffusion policy[1] [6] and the world model is Dreamer-v3 [8]. In both simulation and hardware, we call the verifier twice throughout the task; thus $M = 2$, and we perform calibration at this sequence-level.

Simulation Setup. We use the Robomimic [16] benchmark and study the task of putting the nut on the peg. We chose this task because it requires high-precision, but also the human demonstrations in Robomimic exhibit inherent diversity in *how* the nut is be placed on the peg motions (e.g., placed with handle facing left or right). We train our low-level diffusion policy with 120 demonstrations, 60 placing with its handle facing left, and the other 60 with the handle facing right. We train the world model initially on 600 trajectories: 120 demonstrations, and 480 rollouts from our base policy. We also fine-tune the model with 100 additional rollouts from the interleaved policy and world model prediction scheme.

Real World Setup. We use a Franka Emika robotic manipulator equipped with a Robotiq gripper. Images are perceived from two cameras: a wrist-mounted camera and a third-person camera mounted to the robot's left (see Appendix C). We consider a household manipulation task where a robot needs to pick a green cup and place it into one of two orange bins. The bins are tagged with two labels: left (clean) and right (dirty). Task uncertainty can arise when there is ambiguity in the user's instruction on how to clean up the table (e.g., target location left/right, or tag information of clean/dirty). The base policy is a diffusion policy that controls the end-effector pose and gripper width. We train it using 100 demonstrations, covering two distinct behavioral modes: placing the cup in the left/right bin. The world model is trained on a dataset of 350 trajectories, comprising the 100 expert demonstrations and an additional 250 rollouts from the base policy.

VLM Narration & Verification. We use the same set of VLM models in simulation and in hardware. For narrating the world model imaginations, we use Gemini-3-flash-preview due to its strong video summarization capabilities. For verification, we use Gemini-2.0-flash because it exposes the token logits via the API, necessary for ablations for conformal prediction in Appendix E.

A. Uncertainty-Aware Steering

Calibration Dataset. We collect 80 pairs of initial observation o_0 and language instruction $\mathcal{L}$. Our tasks have multiple phases $M = 2$ and we obtain new observation o_T after the first action sequence is executed, gathering data of both phases as our calibration set. 40 of these instructions are ambiguous, such as *"move the cup to a bin"*, and 40 are straightforward, such as *"can you move it to the dirty bin?"*. Among straightforward instructions, sometimes the policy does not generate any target actions, leading the scenario to be labeled incapable. From the observation, we obtain the behavior narrations for the imagined rollouts of $K = 10$ action samples. After narration, we group semantically identical narrations and randomly select a narration from each group to appear as a multiple choice option. We choose $1 - \epsilon = 0.85$ for calibration.

[1]Although we focus on diffusion models, our framework can be applied to any base policy that represents a stochastic action distribution and from which diverse action samples can be drawn, e.g., a vision-language-action model.

Evaluation Dataset. Similar to our calibration dataset, we maintain a set of 40 samples in the test set sampled from the same distribution as the calibration dataset. During evaluation, we predict the set for each sample and compare that to the ground-truth set. If the ground-truth option(s) lie in the prediction set, we have achieved coverage. If the set size is larger than 1, we need clarification.

Metrics. We define three metrics commonly used in prior CP work to evaluate our calibrated VLM verifier. **Coverage** is the proportion of test samples whose ground-truth option(s) are included in the prediction set. The **Clarification rate** is the proportion of times that the prediction set is larger than 1 and thus the robot asks the human a question. The **Set size** is the size of the prediction set. For coverage, we want to achieve the desired coverage rate $1-\epsilon$ in all three scenarios. For the clarification rate, we want a high clarification rate in ambiguous cases and a low one in the other two. For the prediction set, we want the set size to be 2 for ambiguous cases and 1 for other cases.

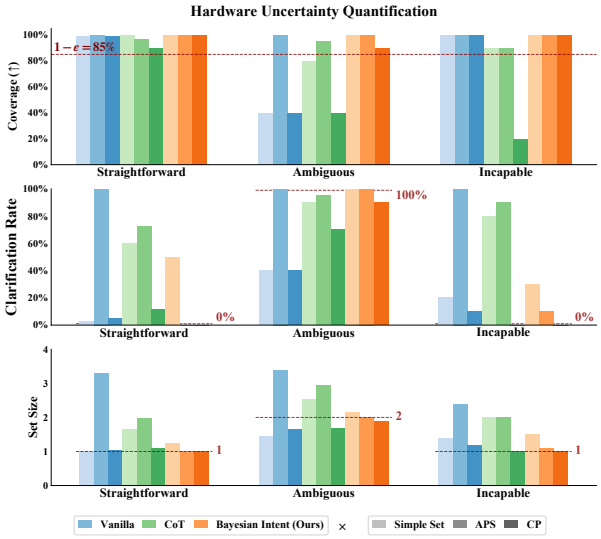


Fig. 3: **Uncertainty Quantification Results: Hardware**. Combination of Vanilla, CoT and Bayesian Intent (Ours) models for UQ. Dashed lines are either the target coverage rate ($1-\epsilon=0.85$), clarification rate, or set size.

1) Our Score Function Balances Coverage & Clarification: **Methods.** We evaluate three different UQ methods: Simple Set, APS [22], CP [21], and three different score functions including Vanilla, CoT and Bayesian Intent (Ours). This results in 3x3 methods we compare. **CP**: we adopt the same conformal prediction scheme from [21] but modify the score function to be Eq 7. **Simple Set**: sorts the options from high scores to low and adds the options until the sum exceeds $1-\epsilon$. **APS** sorts the options from high to low scores and adds the options until the sum exceeds $1-\hat{q}$. **Vanilla** uses the distribution p^{VLM} by asking the model to self-generate the probability scores between 0 and 1. **CoT** shapes p^{VLM} by first prompting the VLM to reason step-by-step and then, conditioned on the reasoning, generates the self-reported probability. **Bayesian Intent** computes intent-conditioned scores p^{VLM} as in Eq. 5.

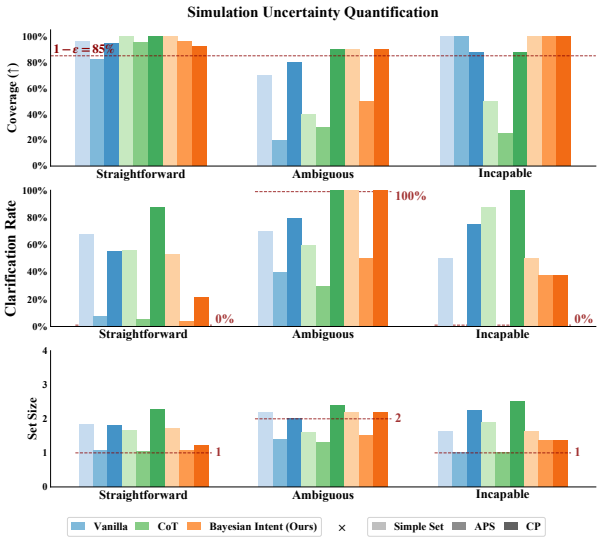


Fig. 4: **Uncertainty Quantification Results: Simulation**. We compare the combination of Vanilla, CoT and Bayesian Intent (Ours) models for UQ. Dashed lines are either the target coverage ($1-\epsilon=0.85$), clarification rate, or set size.

Evaluation. We collect the evaluation dataset in Sec. V-A. When the policy is incapable under straightforward instruction, it is counted as an incapable scenario. Details of the dataset composition are in Appendix E.

Results. Across simulation and hardware, Fig. 4 and Fig. 3 show that our proposed Bayesian intent score function with CP can achieve strong coverage ($\geq 85\%$) while minimizing the clarification rate compared to Vanilla and CoT. Our proposed Bayesian intent score function achieves the best balance between coverage and clarification rate across both tasks. Among hardware UQ methods, CP is more effective in predicting ideal set sizes. In contrast, CoT-based methods perform poorly due to conservative reasoning; they frequently overestimate ambiguity (leading to near 100% clarification rates) or exhibit sharp drops in coverage when set sizes are reduced (APS CoT). Similarly, Vanilla methods struggle with poor calibration, resulting in either overconfidence or severe under-coverage. This indicates that shaping the VLM's uncertainty via factorization leads to better calibration.

2) Uncertainty Improves Policy Steering Performance: Next, we study if our well-calibrated VLM verifier (CP + Bayesian Intent) results in closed-loop performance improvements within our policy steering framework.

Methods. We compare three methods in this section including **Base Policy**, the low-level diffusion policy, **Forewarn** [30], an *uncalibrated* variant of VLM-in-the-loop policy steering, and **UPS w/ Clarification** which uses the calibrated threshold $\hat{q}$ to select the prediction set. When set size >1, it communicates

with user through Q & A and updates its task instructions accordingly to verify again as described in Sec. IV-C.

Evaluation. We take the same dataset of 40 samples as in Sec.V-A1. We compute a confusion matrix where true positive (TP) indicates the selected action sample leads to the correct outcome; true negative (TN) indicates the system properly elected to ask for help, false positive (FP) indicates the selected action actually fails; false negative (FN) indicates the system chose to ask for unnecessary help. We report the success rate as the number of true positives divided by total number of samples. The detailed analysis is in Appendix C and D.

Results. Fig. 5 shows that although Forewarn improves the success rate in both straightforward scenarios and ambiguous scenarios by steering the base policy to better align with the task instructions, it achieves much smaller gain in ambiguous cases in both tasks (e.g., in hardware task improvement in straightforward = 45%, ambiguous = 15%) because the VLM verifier is overconfident. In contrast, UPS + w/ Clarification addresses uncertainty in ambiguous cases, further increasing success rate by 15%. We note that with UQ and clarification, we can already achieve the desired coverage rate of 85% in straightforward cases.

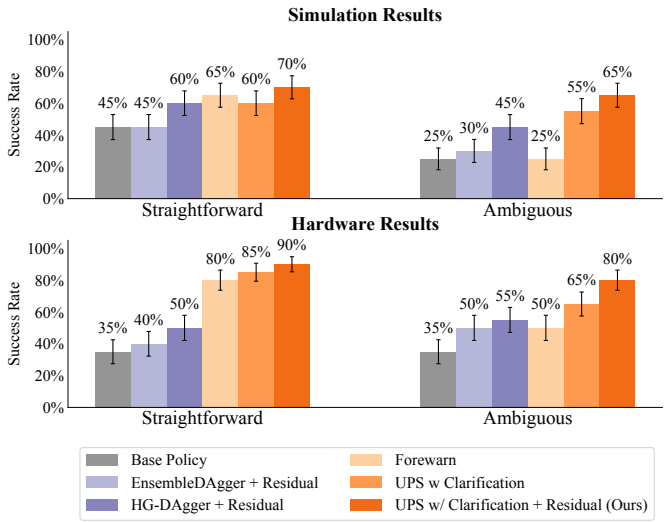


Fig. 5: **Success Rates Pre- and Post-Continual Learning: Hardware and Simulation.** We deploy the robot with 20 straightforward (left) and 20 ambiguous (right) task instructions. We average the success rate over 20 trials for each scenario. Our approach solicits data in a way which maximizes the final success rate after residual policy training, compared to human- and robot-gated baselines.

B. Continual Learning

Finally, we close the loop and evaluate how our uncertainty-aware steering method informs low-level data collection for improving the base policy, and how this influences human intervention rate and re-deployment performance.

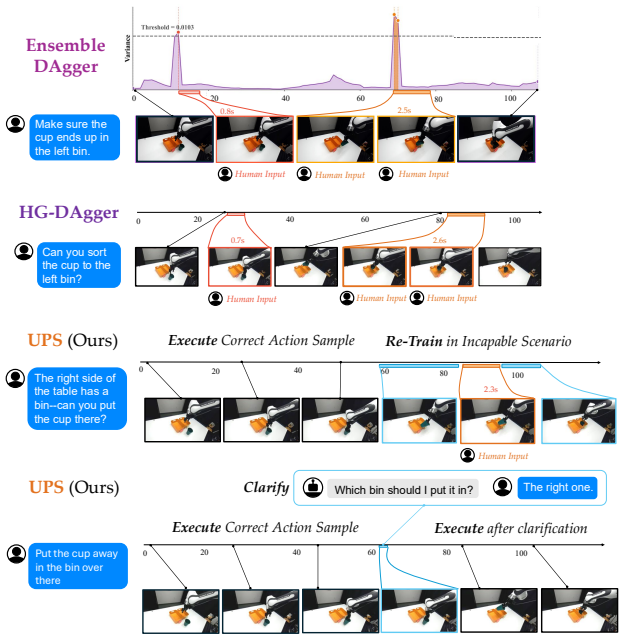


Fig. 6: **Hardware: Robot Asking for Interventions.** In EnsembleDAgger (top), the human is asked for demonstrations whenever the disagreement across an ensemble of base policies exceeds a threshold. In Human-Gated (HG) DAgger (middle), a human monitors the entire trajectory and intervenes whenever the robot's behavior deviates from their intention. Our approach (bottom two rows), enables the robot to ask for "cheap" clarification questions and only requests corrections when no action samples achieve the user's instruction.

1) Semantic-level UQ Minimizes Human Feedback:

Evaluation. Similar to Sec. V-A2, we evaluate the policy for 40 trials where each trial has different initial condition and different language instructions. During intervention, the human corrects the robot by controlling the end-effector and gripper through a teleoperation device (e.g., SpaceMouse). We use the same success rate metric as in Sec. V-A2.

Methods. We compare our proposed high-level semantic guided human interventions **UPS + w/ Clarification + Residual** against two foundational approaches, shown in Fig. 6: **HG-DAgger** [11], where a human intervenes whenever they see fit, and **EnsembleDAgger** [19], which asks for human interventions when the policy ensemble disagreement is high.

Metrics. For each trajectory, we divide the number of human intervention steps by trajectory length, and then we average this normalized rate across all the trajectories as our metrics to measure overall human effort. When our approach asks a clarification question *without* low-level interventions, we count this as one human intervention step since Q&A is easier than physically controlling the robot.

Results. **UPS** has the lowest human intervention rate because it asks for low-level intervention data only when the policy cannot generate a low-level behavior that is "semantically aligned" with the task. Our method achieves significantly

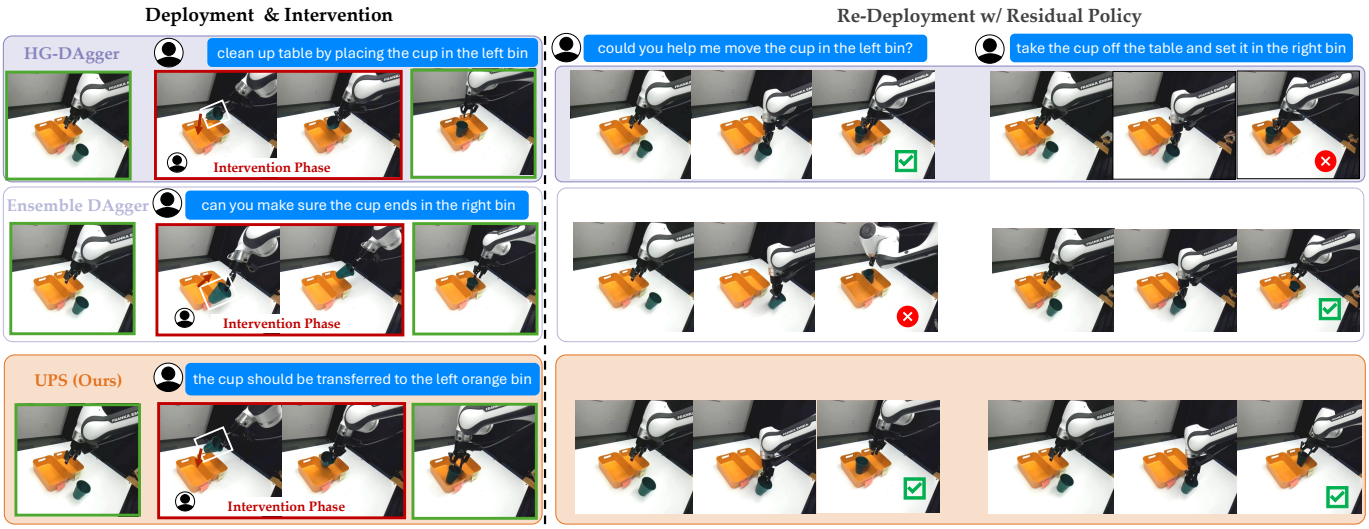


Fig. 7: **Qualitative Results of Intervention & Re-deployment**. On the left, we demonstrate three different strategies to elicit human interventions: HG-DAgger (top), EnsembleDAgger (middle) and UPS (Ours) (bottom). On the right, we demonstrate that our method maintains the multi-modality to be able to place the cup in the left or right bin while other methods suffer from failures and lack of multi-modality.

lower intervention rates than the baselines in both settings. In hardware, UPS requires interventions at a rate of 0.06, compared to 0.1 for HG-DAgger and 0.16 for EnsembleDAgger, whose low-level uncertainty quantification demands even more human assistance. In simulation, the gap widens: UPS has an intervention rate of 0.058, while HG-DAgger has an intervention rate of 0.20 and EnsembleDAgger has 0.27.

2) UPS Improves Re-Deployment Performance:

Residual Policy Training. For fair comparison among different intervention methods, we fix an intervention budget and give all methods the same number of intervention trajectories in their residual learning dataset. We compute the delta action between the intervention action and the base action and train the residual policy π^{residual} with the observation and base action as features, as shown in Appendix F.

Methods. Similar to Sec. V-B1, we learn a residual policy from the intervention data we collected with different methods: **EnsembleDAgger + Residual**, **HG-DAgger + Residual** and **UPS + w/ Clarification + Residual**. We also compare to the base policy without the residual. We use the success rate metric described in Sec. V-A2.

Results. Fig. 5 shows that our residual policy continues to improve from **UPS w/ Clarification** only because it reduces the proportion of incapable scenarios; it obtains a 15% increase in success rate in ambiguous settings in hardware. Across both types of scenarios, our final success rate is 85% in hardware. This empirical result corroborates Theorem 1, demonstrating that the verifier can steer the low-level policy to success in $1-\epsilon$ of scenarios by either executing the right sample or selecting the appropriate strategy (asking clarification questions when ambiguous or eliciting human interventions when incapable). In simulation, we achieve slightly lower coverage due to the challenges of simulating precise contact-rich manipulation tasks with the current world model. On the other hand, both baseline DAgger methods struggle to learn new behaviors intervention data while maintaining the original capabilities as shown in Fig. 7. We hypothesize that this could be because (a) they can't distinguish when the low-level policy has learned diverse strategies at doing a task vs. when it is incapable, and (b) the learned residual policy biases the final policy distribution towards certain behavior modes that are aligned for only some ways of doing the task, or may push the model towards new incapable modes.

VI. Conclusion & Limitations

In this work, we propose an uncertainty-aware policy steering system which rigorously calibrates a verifier's uncertainty over low-level action samples. Our approach also enables the robot to resolve uncertainty with high-level clarifications or low-level re-training to achieve overall success rate in $1-\epsilon$ of scenarios. Since we focus on uncertainty quantification of the VLM verifier, we assumed the imaginations from the world model always match future outcomes and the behavior narrations are correct and complete. Interesting future work incorporates the uncertainty of the world model (and narrations) [17], for an even more robust system.

References

[1] Moises Andrade, Joonhyuk Cha, Brandon Ho, Vriksha Srihari, Karmesh Yadav, and Zsolt Kira. Let's think in two steps: Mitigating agreement bias in mllms with self-grounded verification. In *International Conference on Learning Representations (ICLR)*, 2026.

[2] Anastasios N Angelopoulos, Stephen Bates, et al. Conformal prediction: A gentle introduction. *Foundations and trends® in machine learning*, 16(4):494–591, 2023.

[3] Zechen Bai, Pichao Wang, Tianjun Xiao, Tong He, Zongbo Han, Zheng Zhang, and Mike Zheng Shou. Hallucination of multimodal large language models: A survey. *arXiv preprint arXiv:2404.18930*, 2024.

[4] Chris L Baker, Joshua B Tenenbaum, and Rebecca R Saxe. Goal inference as inverse planning. In *Proceedings of the annual meeting of the cognitive science society*, volume 29, 2007.

[5] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0 : A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[6] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

[7] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

[8] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *Nature*, 2023.

[9] Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S Brown, and Ken Goldberg. Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning. In *5th Annual Conference on Robot Learning*, 2021.

[10] Yunfan Jiang, Chen Wang, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Transic: Sim-to-real policy transfer by learning from online correction. In *Conference on Robot Learning*, pages 1691–1729. PMLR, 2025.

[11] Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083. IEEE, 2019.

[12] Jacky Kwok, Christopher Agia, Rohan Sinha, Matt Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. Robomonkey: Scaling test-time sampling and verification for vision-language-action models. *arXiv preprint arXiv:2506.17811*, 2025.

[13] Justin Lidard, Hang Pham, Ariel Bachman, Bryan Boateng, and Anirudha Majumdar. Risk-calibrated human-robot interaction via set-valued intent prediction. *Robotics: Science and Systems*, 2024.

[14] Huihan Liu, Yu Zhang, Vaarij Betala, Evan Zhang, James Liu, Crystal Ding, and Yuke Zhu. Multi-task interactive robot fleet learning with visual world models. In *8th Annual Conference on Robot Learning*, 2024.

[15] Yuxuan Liu, Tianchi Yang, Shaohan Huang, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. Calibrating llm-based evaluator. In *Proceedings of the 2024 joint international conference on computational linguistics, language resources and evaluation (lrec-coling 2024)*, pages 2638–2656, 2024.

[16] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

[17] Zhiting Mei, Tenny Yin, Micah Baker, Ola Shorinwa, and Anirudha Majumdar. World models that know when they don't know: Controllable video generation with calibrated uncertainty. *arXiv preprint arXiv:2512.05927*, 2025.

[18] Zhiting Mei, Christina Zhang, Tenny Yin, Justin Lidard, Ola Shorinwa, and Anirudha Majumdar. Reasoning about uncertainty: Do reasoning models know when they don't know? *arXiv preprint arXiv:2506.18183*, 2025.

[19] Kunal Menda, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Ensembledagger: A bayesian approach to safe imitation learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5041–5048. IEEE, 2019.

[20] James F Mullen and Dinesh Manocha. Lbap: Improved uncertainty alignment of llm planners using bayesian inference. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 18716–18723. IEEE, 2025.

[21] Allen Z Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, et al. Robots that ask for help: Uncertainty alignment for large language model planners. *Conference on Robot Learning*, 2023.

[22] Yaniv Romano, Matteo Sesia, and Emmanuel Candes. Classification with valid and adaptive coverage. *Advances in neural information processing systems*, 33: 3581–3591, 2020.

[23] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[24] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

[25] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[26] Pablo Valle, Chengjie Lu, Shaukat Ali, and Aitor Arrieta. Evaluating uncertainty and quality of visual language action-enabled robots. *arXiv preprint arXiv:2507.17049*, 2025.

[27] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer, 2005.

[28] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[29] Yilin Wu, Anqi Li, Tucker Hermans, Fabio Ramos, Andrea Bajcsy, and Claudia P'erez-D'Arpino. Do what you say: Steering vision-language-action models via runtime reasoning-action alignment verification. *arXiv preprint arXiv:2510.16281*, 2025.

[30] Yilin Wu, Ran Tian, Gokul Swamy, and Andrea Bajcsy. From foresight to forethought: Vlm-in-the-loop policy steering via latent alignment. *Robotics: Science and Systems*, 2025.

[31] Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi Fan, et al. Self-improving vision-language-action models with data generation via residual rl. *arXiv preprint arXiv:2511.00091*, 2025.

[32] Michelle D. Zhao, Henny Admoni, Reid Simmons, Aaditya Ramdas, and Andrea Bajcsy. Conformalized interactive imitation learning: Handling expert shift and intermittent feedback. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=Ym2RNPX6la.

[33] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. In *Forty-second International Conference on Machine Learning*, 2025.

[34] Thomas P Zollo and Richard Zemel. Confidence calibration in vision-language-action models. *arXiv preprint arXiv:2507.17383*, 2025.

APPENDIX

A. Proof of Coverage Guarantees

To maintain formal coverage guarantees despite temporal dependencies, we follow the same proof style as in [21] and perform sequence-level calibration using the minimum score over phases.

Proof: Let the calibration dataset be *i.i.d.* sequences of $(\bar{x}^n, \bar{y}^n)$, $(\bar{x}^n, \bar{y}^n) \overset{\text{i.i.d.}}{\sim} \mathcal{D}$:

$$D_{\text{calib}} = \{\{(x_i^n, \mathcal{Y}_i^n)\}_{i=0}^M\}_{n=1}^N = \{(\bar{x}^n, \bar{y}^n)\}_{n=1}^N,$$

where $\bar{x}^n$ is a sequence of query points and $\bar{y}^n$ assigns to each $x_i^n \in \bar{x}^n$ a set of correct labels $\mathcal{Y}_i^n \subseteq \mathcal{Y}$. For any sequence $(\bar{x}, \bar{y})$, define the per-point "multi-option" nonconformity score

$$s(x_i, \mathcal{Y}_i) \;:=\; 1 - \min_{y \in \mathcal{Y}_i} p^{\text{VLM}}(y \mid x_i), \tag{1}$$

and define the sequence-level nonconformity score (equivalently to the theorem's definition)

$$\kappa(\bar{x}, \bar{y}) \;:=\; 1 - \min_{x_i \in \bar{x}} \; \min_{y \in \mathcal{Y}_i} p^{\text{VLM}}(y \mid x_i) \;=\; \max_{x_i \in \bar{x}} s(x_i, \mathcal{Y}_i). \tag{2}$$

Let $\kappa^n := \kappa(\bar{x}^n, \bar{y}^n)$ be the n-th calibration score for $n \in [N]$, and let $\hat{q}$ be the split-conformal quantile of $\{\kappa^n\}_{n=1}^N$ at level $1-\varepsilon$, i.e. the $\lceil (N+1)(1-\varepsilon) \rceil$-th order statistic (standard split conformal rule).

a) Step 1: The standard conformal prediction guarantee for i.i.d. samples:

Draw an independent test sequence

$$(\bar{x}^{\text{test}}, \bar{y}^*) \sim \mathcal{D}, \qquad \bar{x}^{\text{test}} = \{x_i^{\text{test}}\}_{i=0}^M, \;\; \bar{y}^* = \{\mathcal{Y}_i^*\}_{i=0}^M,$$

and define its sequence-level score $\kappa^{\text{test}} := \kappa(\bar{x}^{\text{test}}, \bar{y}^*)$. Since $(\bar{x}^1, \bar{y}^1), \ldots, (\bar{x}^N, \bar{y}^N), (\bar{x}^{\text{test}}, \bar{y}^*)$ are *i.i.d.* and $\kappa(\cdot, \cdot)$ is a deterministic function of a sequence, the $(N+1)$ random variables $(\kappa^1, \ldots, \kappa^N, \kappa^{\text{test}})$ are *i.i.d.*. Hence, by the standard split conformal quantile guarantee,

$$\Pr\big(\kappa^{\text{test}} \le \hat{q}\big) \;\ge\; 1 - \varepsilon. \tag{3}$$

b) Step 2: Sequence-level calibration guarantees the per-point coverage as shown in Claim 1 in [21]:

Define the conformal prediction set for any query point x in any sequence by

$$C(x) \;:=\; \{y \in \mathcal{Y} :\; 1 - p^{\text{VLM}}(y \mid x) \le \hat{q}\}. \tag{4}$$

From Definition 2, we have $\kappa^{\text{test}} = \max_{x_i^{\text{test}} \in \bar{x}^{\text{test}}} s(x_i^{\text{test}}, \mathcal{Y}_i^*)$ Since from Eq. 3, we have

$$\Pr\big(\kappa^{\text{test}} \le \hat{q}\big) \;\ge\; 1 - \varepsilon.$$

Then

$$\Pr\left(\max_{x_i^{\text{test}} \in \bar{x}^{\text{test}}} s(x_i^{\text{test}}, \mathcal{Y}_i^*) \le \hat{q}\right) \;\ge\; 1 - \varepsilon.$$

Given that

$$\forall x^{\text{test}} \in \bar{x}^{\text{test}}, s(x^{\text{test}}, \mathcal{Y}_i^*) \le \max_{x_i^{\text{test}} \in \bar{x}^{\text{test}}} s(x_i^{\text{test}}, \mathcal{Y}_i^*)$$

We have

$$\forall x^{\text{test}} \in \bar{x}^{\text{test}}, \Pr\big(s(x_i^{\text{test}}, \mathcal{Y}_i^*) \le \hat{q}\big) \;\ge\; 1 - \varepsilon \tag{5}$$

This indicates if the test sequence $\bar{x}^{\text{test}}$ satisfies the guarantee, then any query point x^{test} in the test sequence also satisfies the guarantee.

c) Step 3: Minimum score over true option sets guarantees all true options are included:

From Definition 1 and Eq. 5, we know $\forall x^{\text{test}} \in \bar{x}^{\text{test}}$,

$$\Pr\left(1 - \min_{y \in \mathcal{Y}^*} p^{\text{VLM}}(y \mid x_i^{\text{test}}) \le \hat{q}\right) \;\ge\; 1 - \varepsilon$$

Given that

$$\forall y' \in \mathcal{Y}_i^*, 1 - p^{\text{VLM}}(y' \mid x_i^{\text{test}}) \le 1 - \min_{y \in \mathcal{Y}_i^*} p^{\text{VLM}}(y \mid x_i^{\text{test}})$$

We have $\forall y' \in \mathcal{Y}_i^*$,

$$\Pr\left(1 - p^{\text{VLM}}(y' \mid x_i^{\text{test}}) \le 1 - \min_{y \in \mathcal{Y}_i^*} p^{\text{VLM}}(y \mid x_i^{\text{test}}) \le \hat{q}\right) \ge 1 - \varepsilon$$

Therefore, combined with Definition 4, we get

$$\Pr\big(\mathcal{Y}_i^* \subseteq C(x_i^{\text{test}})\big) \;\ge\; 1 - \varepsilon,$$

which proves the claim.

We have proved that $\Pr\big(\mathcal{Y}_i^* \subseteq C(x_i^{\text{test}})\big) \ge 1 - \epsilon$, where the probability is taken over the random draw from the calibration set D_{calib} and the test pair $\big(x_i^{\text{test}}, \mathcal{Y}_i^*\big)$; that is, the result is a marginal coverage guarantee. Consequently, achieving the target coverage for each new test point would, in principle, require a freshly sampled calibration set. However, following [21], we use the dataset-conditional guarantee — conditioning on the realized calibration dataset — which can be applied to new test samples without re-calibration.

B. Aside on our score function design

Prior work [21] uses the *max* score to get an existence guarantee in Eq 6 and is designed to ensure the prediction set contains *at least one* acceptable choice and then commits to a single selected option. We aim to include *all* suitable options in the prediction set by using a more conservative score that minimizes over options to achieve a completeness guarantee (Eq. 7). From a user perspective, this avoids guessing the user's unstated intent and instead displays the set of plausible choices for quick disambiguation.

$$\{\tilde{\kappa}_i^n = 1 - \min_{1 \le i \le h} \max_{y \in \mathcal{Y}_i^n} p^{\text{VLM}}(y \mid x_i^n)\}_{n=1}^N \tag{6}$$

KnowNo maximizes over multiple ground-truth options

$$\{\tilde{\kappa}_i^n = 1 - \min_{1 \le i \le h} \min_{y \in \mathcal{Y}_i^n} p^{\text{VLM}}(y \mid x_i^n)\}_{n=1}^N \tag{7}$$

Ours minimizes over multiple ground-truth options

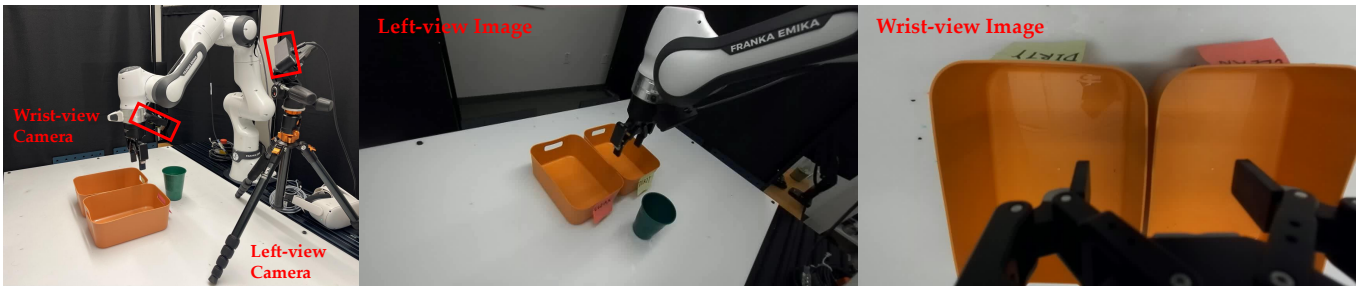


Fig. 8: **Hardware Setup.** We demonstrate our hardware environment setup with a Franka Emika Panda arm and two Zed cameras (left image). In the middle, we show the left-view image captured by a Zed 2i camera and on the right, we show the wrist-view image captured by a Zed M camera.

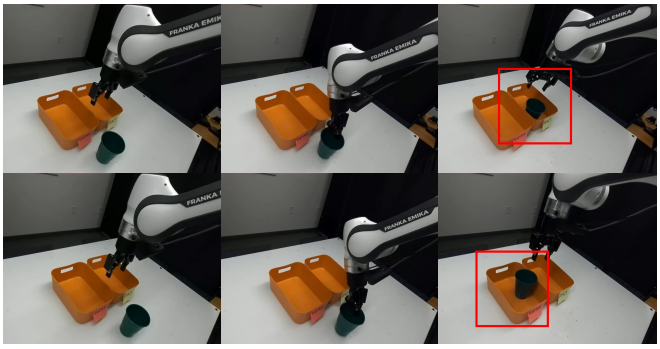


Fig. 9: **Hardware Examples of the Task.** We show two different ways of achieving the task of placing the cup in the bin. The top row shows the cup placed in the left bin while the bottom row shows the cup placed in the right bin.

C. Real Robot (Hardware)

Hardware setup. Fig. 8 shows the setup for our real-world task of placing a cup into a bin. On the table, a green cup is positioned in front of two orange bins: the left bin is marked "clean" with a pink tag, and the right bin is marked "dirty" with a yellow tag. Both the left-view and wrist-view images have dimensions of 1920×1080. The height is zero-padded to match the width, and the resulting square images are resized to 256×256 for input to the policy and world model. To achieve the goal of placing the cup in a bin, the robot can complete the task in two ways by selecting either the left or right bin, as shown in Fig. 9.

Task Instructions. We divide the different instructions a user might give for the robot to complete the task into two categories: 1) *straightforward* scenarios; 2) *ambiguous* scenarios. In Table II and Table III, we list all task instructions used to construct the calibration dataset. The instructions at test-time are randomly sampled from this same set of instructions.

Prompts. In our proposed framework and corresponding baselines, we use a VLM for a variety of purposes. Therefore, we list all the prompts we use in our experiments.

Specifically, to enable the VLM to serve as a behavior narration translator $\mathcal{T}^{\text{VLM}}$, we use the prompt listed in Fig. 15 during the grasping phase and Fig. 16 during the placing phase in our hardware settings. For VLM verification in our Forewarn baseline, we follow prior work [30] by directly prompting the VLM to select an option. The prompts are listed in Fig. 17.

We also include the prompts for different score functions evaluated in Sec. V-A1. For the **Vanilla** baseline, the prompts are listed in Fig. 25. For the **CoT** baseline, we include the two-step prompts that reason first about the ambiguity and then, based on this reasoning, select the option and generate the scores. For the hardware setting, the two-step prompts are included in Fig. 23 and Fig. 24. For our **Bayesian Intent** method, there are three steps: 1) generate the set of possible intents; 2) evaluate each intent's probability among the set; 3) evaluate each behavior mode's probability under each specific intent. In the first step, we use the prompt in Fig. 18. Next, we generate intent probability with the prompt in Fig. 19. Finally, we infer behavior probability with the prompt in Fig. 22.

In addition to translation and verification, the VLM generates clarification questions and automatically updates user instructions based on the responses. The prompts for this clarification phase are the same for hardware and simulation. We use the prompt in Fig. 21 to ask clarification questions and the prompt in Fig. 20 to update instructions once we receive the user's answer.

Results Breakdown. To provide a detailed analysis of our approach and related baselines regarding policy steering performance, we include a breakdown of the True Positive Rate (TP), True Negative Rate (TN), False Positive Rate (FP) and False Negative Rate (FN) in Table I. The success rate metrics shown in Fig. 5 display the number of True Positives divided by the total number of trials. As shown in Fig. I, all policy steering methods significantly reduce the TN compared to the base policy and DAgger variants, indicating the effectiveness of generation and verification. Among those steering methods, our UPS w/ Clarification can already significantly improve the TP and reduce the FP rate in ambiguous situations because the VLM verifier is more calibrated with conformal prediction and our proposed Bayesian Intent Score Function. With Residual Learning, UPS w/ Clarification + Residual can further reduce the TN and improve the TP, which is more obvious in ambiguous scenarios. This detailed analysis demonstrates

Method	Straightforward				Ambiguous			
	TP	TN	FP	FN	TP	TN	FP	FN
Base Policy	0.35	0.65	–	–	0.35	0.65	–	–
EnsembleDagger + Residual	0.40	0.60	–	–	0.50	0.50	–	–
HG-Dagger + Residual	0.50	0.50	–	–	0.55	0.45	–	–
Forewarn	0.80	0.10	0.10	0.0	0.50	0.0	0.40	0.10
UPS w Clarification	0.85	0.05	0.05	0.05	0.65	0.25	0.10	0.0
UPS w Clarification + Residual (Ours)	**0.90**	0.0	0.10	0.0	**0.80**	0.0	0.20	0.0

TABLE I: **Hardware Policy Steering Results Breakdown**. We include the True Positive Rate (TP), True Negative Rate (TN), False Positive Rate (FP) and False Negative Rate (FN) in the table as an extension to the results we show in Fig. 5. For straightforward and ambiguous scenarios, the reported success rate is averaged across 20 trials, each with a unique task instruction.

Category	User Intent	Task Instruction
Straightforward	left	"on the left side, there is a bin—place the cup there"
	left	"the cup should end up in the left bin"
	left	"take the cup off the table and finish by putting it in the left bin"
	left	"make the left bin the final location for the cup"
	left	"ensure the cup is placed inside the left bin"
	left	"please place the cup in the left bin"
	left	"the cup should be transferred to the left orange bin"
	left	"clean up the table by placing the cup in the left bin"
	left	"grasp the cup and place it in the left bin"
	left	"could you ensure the left bin is the final location for the cup"
	right	"on the right side, there is a bin—send the cup there"
	right	"the cup should end up in the right bin"
	right	"take the cup off the table and finish by setting it in the right bin"
	right	"make the right bin the final location for the cup"
	right	"ensure the cup is inside the right bin"
	right	"could you move the cup to the bin on the right side?"
	right	"can you make sure the right-hand container is where the cup ends up?"
	right	"the bin on the right needs the cup inside it; could you handle that?"
	right	"the right side of the table has a bin—can you put the cup there?"
	right	"could you ensure the cup is placed in the bin to your right?"
	dirty (right)	"since the cup is dirty, it belongs in the dirty bin"
	dirty (right)	"do not store it with clean items—send the cup to the dirty bin"
	dirty (right)	"treat this as a dirty item and place the cup accordingly"
	dirty (right)	"the destination depends on its condition: dirty cup → dirty bin"
	dirty (right)	"put the cup wherever dirty cups are supposed to go (the matching bin)"
	dirty (right)	"could you put the cup in the dirty bin because it is soiled?"
	dirty (right)	"this cup isn't clean; can you make sure it goes in the dirty container?"
	dirty (right)	"the dirty bin is the correct spot for this cup; could you put it there?"
	dirty (right)	"since we have a dirty cup here, can you move it to the dirty bin?"
	dirty (right)	"could you match this cup's condition by putting it in the dirty bin?"
	clean (left)	"since the cup is clean, it belongs in the clean bin"
	clean (left)	"do not mix it with dirty items—send the cup to the clean bin"
	clean (left)	"treat this as a clean item and place the cup accordingly"
	clean (left)	"the destination depends on its condition: clean cup → clean bin"
	clean (left)	"put the cup wherever clean cups are supposed to go (the matching bin)"
	clean (left)	"could you put the cup in the clean bin since it is already washed?"
	clean (left)	"this cup is clean; can you ensure it ends up in the clean container?"
	clean (left)	"the clean bin is the right place for this; could you set it inside?"
	clean (left)	"because this is a clean cup, can you move it to the clean bin?"
	clean (left)	"could you place the cup in the clean bin so it stays sanitary?"

TABLE II: **Straightforward Hardware Task Instructions.** 40 straightforward task instructions used for the calibration dataset are listed here. During testing, we randomly sample 20 instructions from the set and obtain new options when sampling from the policy.

Category	User Intent	Task Instruction
Ambiguous	left or right	“can you deal with the cup and put it away?”
	left or right	“the cup should not be on the table anymore—put it somewhere appropriate”
	left or right	“tidy this up: move the cup into a bin”
	left or right	“pick up the cup and store it in a container”
	left or right	“put the cup away in the bin over there”
	left or right	“could you move the cup out of the working area?”
	left or right	“take care of the cup and put it in a bin, please”
	left or right	“can you get the cup into one bin?”
	left or right	“the cup has some water—could you put it away somewhere?”
	left or right	“there is water in the cup; can you relocate it to a bin?”
	left or right	“just put the cup away”
	left or right	“could you remove the cup from the table and place it in a container?”
	left or right	“put the cup in the orange bin”
	left or right	“can you put the cup into the orange container?”
	left or right	“could you put the cup in a container so it is not in the way?”
	left or right	“store the cup in the bin, please”
	left or right	“clear the space by moving the cup into a bin”
	left or right	“the cup goes in a bin—can you handle it?”
	left or right	“could you move the cup to a bin and clean up the table?”
	left or right	“can you put the cup somewhere it belongs, like a bin?”
	left or right	”the cup should be settled in one bin”
	left or right	“can you place the cup into a bin to finish cleaning?”
	left or right	“put the cup into a bin to organize the scene”
	left or right	“remove the cup and store it in a bin-like container”
	left or right	“put the cup into a bin and keep the area neat”
	left or right	“move the cup into any bin so the table is usable”
	left or right	“take the cup and put it in a bin for now”
	left or right	“could you put the cup into a bin so it’s not in the working zone?”
	left or right	“place the cup into a bin and return the surface to empty”
	left or right	“the cup should be contained—put it in a bin”
	left or right	“put the cup in a bin so it won’t spill on the table”
	left or right	“since the cup has liquid, move it into a bin”
	left or right	“move the cup with water into a bin to avoid mess”
	left or right	“put the cup somewhere secure, like inside a bin”
	left or right	“could you store the cup inside a bin and clear the workspace?”
	left or right	“take the cup away from the table and place it in a bin”
	left or right	“can you put the cup away in one of the bins over there?”
	left or right	“put the cup in a bin over there and tidy up”
	left or right	“move the cup into a bin so the table is clean”
	left or right	“the cup needs to be put away—use one of the bins”

TABLE III: **Ambiguous Hardware Task Instructions.** 40 ambiguous task instructions used for the calibration dataset are listed here. During testing, we randomly sample 20 instructions from the set and obtain new options with the new action samples from the base policy.

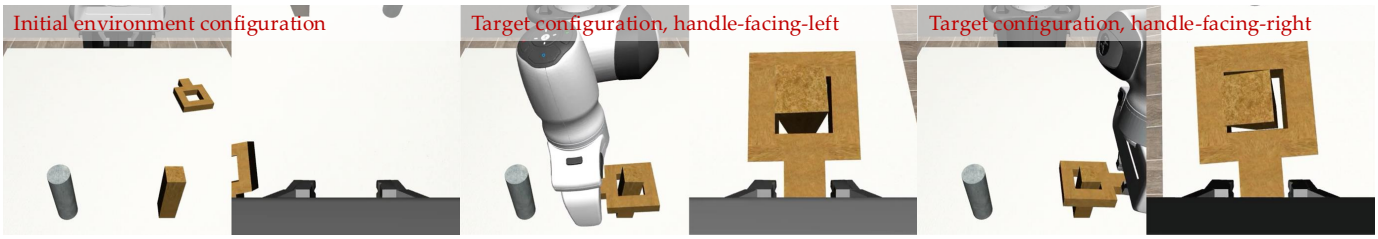


Fig. 10: **Simulation Setup.** We demonstrate our simulated task setup using a Franka Emika Panda arm. The leftmost two images show the initial environment setup, while the right four images show the target configurations for the two behavior modes, as labeled. In each image pair, the left image provides the third-person camera view and the right image provides the wrist view.

Method	Straightforward				Ambiguous			
	TP	TN	FP	FN	TP	TN	FP	FN
Base Policy	0.45	0.55	–	–	0.25	0.75	–	–
EnsembleDagger + Residual	0.45	0.55	–	–	0.30	0.70	–	–
HG-Dagger + Residual	0.60	0.40	–	–	0.45	0.55	–	–
Forewarn	0.65	0.20	0.15	0.0	0.0	0.25	0.15	0.60
UPS w Clarification	0.60	0.20	0.10	0.10	0.55	0.25	0.20	0.0
UPS w Clarification + Residual (Ours)	**0.70**	0.20	0.05	0.05	**0.65**	0.10	0.15	0.10

TABLE IV: **Simulation Policy Steering Results Breakdown**. We include the True Positive Rate (TP), True Negative Rate (TN), False Positive Rate (FP) and False Negative Rate (FN) in the table as an extension to the results we show in Fig. 5. For straightforward and ambiguous scenarios, the reported success rate is averaged across 20 trials respectively, where each trial has an unique task instruction.

that by calibrating high-level semantic uncertainty with low-level action infeasibility together and selectively choosing an appropriate resolution strategy, we can steer the policy to better achieve user-desired goals, especially when the instruction is ambiguous.

D. Simulation

Simulation Setup. We use the Robomimic NutAssemblySquare environment for our simulated task. Initially, a square nut with a handle protruding from one side lies flat on the table, while a square peg stands upright. The goal is to pick up the nut and place it onto the peg. We focus on two behavior modes to complete this task: placing the nut with its handle facing left versus placing it with the handle facing right, as shown in Fig. 10.

Our observation data consists of two simulated camera views: a wrist-mounted camera on the robot and a third-person camera providing an angled top-down view of the table and robot. Both cameras render at 96×96 resolution for diffusion policy training and deployment. For world-model training and deployment, we downsample these to 64×64, then re-upsample our decoded observations to 96×96 before passing them to the diffusion policy during interleaving.

Task Instruction. We divide the different instructions that a user might give to the robot to complete the task into two categories: 1) *straightforward* instructions, listed in Table VI; 2) *ambiguous* instructions, listed in Table VII. We randomly sample 40 instructions of each category to make up the calibration set, and the remaining instructions form the test set.

Prompts. In our proposed framework and corresponding baselines, we use a VLM for a variety of purposes. Therefore, we list all the prompts we use in our experiments.

Specifically, to enable the VLM to serve as a behavior narration translator $\mathcal{T}^{\text{VLM}}$, we use the prompt listed in Fig. 26 during the grasping phase and Fig. 27 during the placing phase in our simulation setting. For VLM verification for our Forewarn baseline, we follow prior work [30] by directly prompting the VLM to select an option. The prompts are listed in Fig. 28.

We also include the prompts for different score functions evaluated in Sec. V-A1. For the **Vanilla** baseline, the prompts are listed in Fig. 34. For the **CoT** baseline, we include the two-step prompts that reason first about the ambiguity and then, based on this reasoning, select the option and generate the scores. For the simulation setting, the two-step prompts are included in Fig. 31 and Fig. 32. For our **Bayesian Intent** method, there are three steps: 1) generate the set of possible intents; 2) evaluate each intent's probability among the set; 3) evaluate each behavior mode's probability under each specific intent. In the first step, we use the prompt in Fig. 29. Next, we generate intent probability with the prompt in Fig. 30. Finally, we infer behavior probability with the prompt in Fig. 33.

In addition to translation and verification, the VLM generates clarification questions and automatically updates user instructions based on the responses. The prompts for this clarification phase are the same as the hardware setting.

Results Breakdown. Table IV demonstrates the detailed results of all the methods evaluated in Sec. V-A2 and Sec. V-B2, including True Positive Rate (TP), True Negative Rate (TN), False Positive Rate (FP), False Negative Rate (FN). Similar to the trend we observed in hardware experiments, policy steering methods significantly reduce the TN compared to base policy and Dagger variants. In ambiguous scenarios, UPS w/ Clarification significantly improves the TP by 55% through quantifying uncertainty with conformal prediction and further clarifying user intent. With Residual Learning, our method further reduces the TN and FP to improve the TP and overall success rate.

E. Calibration Details

For both hardware and simulation, our calibration dataset contains 80 instructions and our test dataset contains 40 instructions, with each split evenly between ambiguous and straightforward instructions. For each instruction, we sample $K = 10$ action trajectories for phase 1 (grasping) and translate them to text using our scheme from Sec IV. We repeat this process for phase 2 (placing), then combine each instruction with its phase 1 and phase 2 narrations to form a complete scenario.

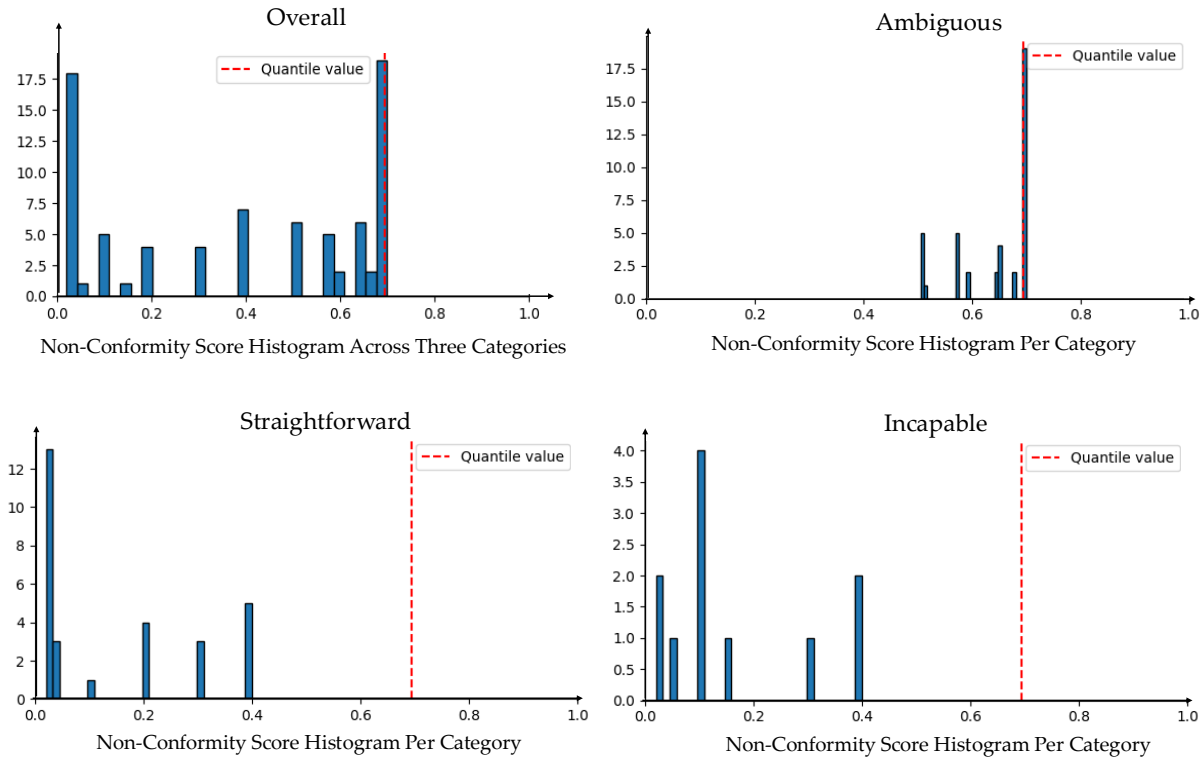


Fig. 11: **Hardware Histogram for Empirical Quantile.** We list the overall histogram with all three categories combined, as well as per category histograms. The red vertical line indicates the $\hat{q}$ value based on user-desired coverage rate $1 - \varepsilon$.

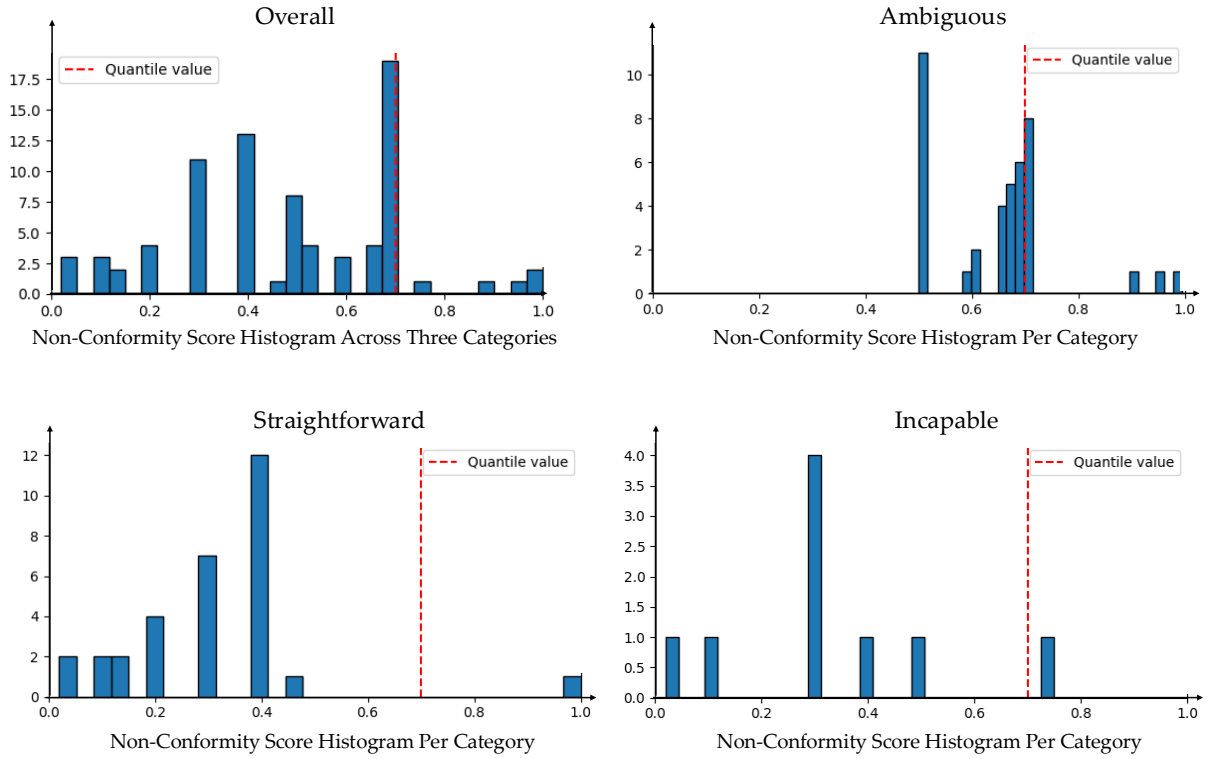


Fig. 12: **Simulation Histogram for Empirical Quantile.** We list the overall histogram with all three categories combined, as well as per category histograms. The red vertical line indicates the $\hat{q}$ value based on user-desired coverage rate $1 - \varepsilon$.

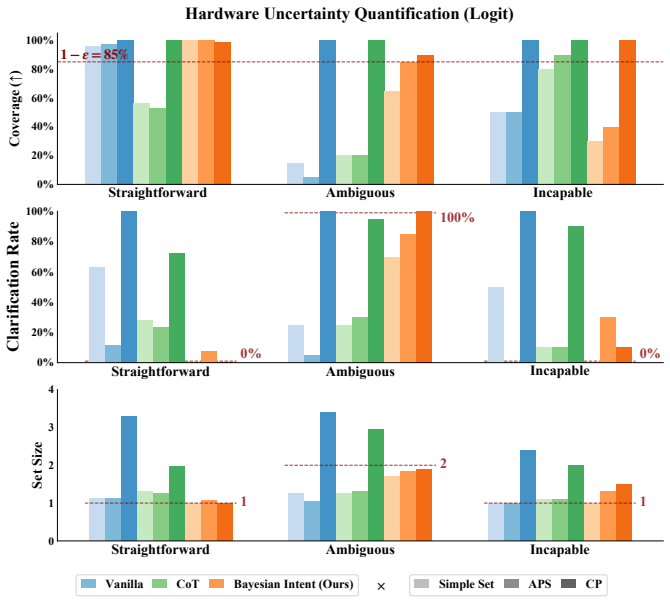


Fig. 13: **Hardware Uncertainty Quantification with Logits**. We conduct an ablation study where we directly use the softmax over the logits of the first generated token as the score rather than the self-generated score. Empirically, we find that, compared to Fig. 3, the self-generated score is a better score function across all scenarios.

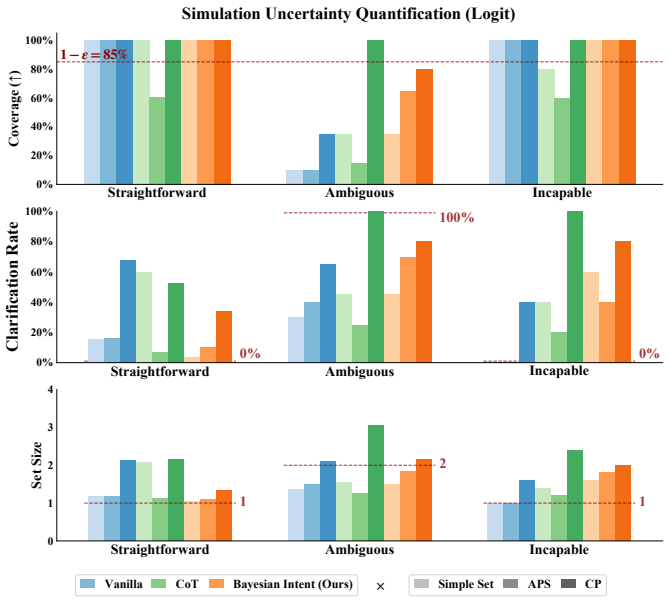


Fig. 14: **Simulation Uncertainty Quantification with Logits**. Similar to hardware experiments, we compare the results of the softmax of logits as the score with self-generated score results in Fig. 4. None of the methods with logits can achieve the desired coverage while balancing the clarification rate and reaching the desired prediction set size.

Among scenarios with **straightforward** instructions, the incapability rates (where no narrations achieve the user's instruction) are as follows. In simulation: calibration scenarios show $0/40$ grasping failures and $9/40$ placing failures, while test scenarios show $0/20$ grasping failures and $5/20$ placing failures. In hardware: calibration scenarios show $2/40$ grasping failures and $10/40$ placing failures, while test scenarios show $1/20$ grasping failures and $5/20$ placing failures.

Results: Nonconformity Score Distribution. For the histograms depicted in Figs. 12 and 11, low nonconformity scores are desirable, as this would indicate our score function correctly assigns high probability mass to the true label. In both hardware and simulation, examining the distribution of non-conformity scores for the calibration set reveals that the majority of samples for both the straightforward and incapable cases have nonconformity scores less than 0.5 (with all but 3 samples meeting this threshold). In the hardware setting specifically, most straightforward and ambiguous samples have nonconformity scores less than 0.2. This indicates that the true label—which is unambiguous in these cases—receives more than 50% probability mass, demonstrating that our score function performs well in these scenarios.

In the ambiguous cases, on the other hand, we see the vast majority of samples (all but 3 samples) having nonconformity scores ranging between 0.5 and 0.7. Recall that in this instance, we take the maximum nonconformity score over two possible valid true labels, so it is not possible for any sample to have a nonconformity score under 0.5 as this would imply that both valid options are assigned greater than 50% probability mass. Therefore, these results imply that across both simulation and hardware, both valid true labels receive at least 30% probability mass, indicating that our score function appropriately distributes probability between the two valid options rather than erroneously favoring one over the other. Together, these calibration results demonstrate that our score function behaves as intended across all three case types.

Results: Uncertainty Quantification with Logits. We compare the uncertainty quantification results from using the softmax of the logits from the VLM in Figs. 14 and 13 to those that ask the VLM to directly self-generate probability score for each option in Figs. 4 and 3, in simulation and hardware. For the score, we compute a softmax function over the logits of the first token (e.g., A, B, C, D) with a temperature of 4. We find that the score functions derived from the softmax over the logits are often biased towards certain options, making other valid options unlikely to be included in the prediction set. Therefore, you can see very low coverage rate for **Vanilla** and **CoT** methods in ambiguous cases compared to self-generated probability scores. Other methods have the desired coverage in ambiguous cases but have larger clarification rate in straightforward and incapable cases because the $\hat{q}$ is increased to cover ambiguous cases. Only the CP with Bayesian Intent (Ours) exhibits good performance across all scenarios in the hardware setting but it is still worse than the performance of self-generated probability scores.

F. Residual Policy

Data Collection. For all interactive imitation learning approaches, when an intervention is deemed necessary (by the human for HG-DAgger or UPS, or by the robot for EnsembleDAgger), the base policy relinquishes control to the user, who teleoperates the robot via SpaceMouse before handing back authority. During human intervention, we continue to run the base policy without deploying its actions, computing $\Delta a = a_{\text{human}} - a_{\text{base}}$ at each timestep. Intervention labels are set to 1 during human operation and 0 otherwise.

Architecture. We separately train a classifier (to predict intervention labels) and a residual policy (to predict residual actions). Both models share the same input features: the observation (camera images and proprioceptive states) and the base policy action at each timestep. All inputs are normalized to the range of $(-1, 1)$, including residual action labels for the residual policy.

Both models use the diffusion policy observation encoder, where each camera image is processed through its own copy of a modified ResNet-18 module. These outputs are concatenated with proprioceptive data and passed through a 2-layer MLP with 512 hidden dimensions [6]. All encoder weights are initialized from the base diffusion policy checkpoint but remain trainable throughout. The base policy action is processed through a single linear layer before being concatenated with the observation encoder output.

For the classifier, this concatenated representation passes through a linear layer to produce a 1-dimensional output, followed by a sigmoid activation to obtain probabilities. We train using binary cross-entropy loss, weighting positive samples to address class imbalance.

For the residual policy, the representation is passed through an action predictor MLP head to produce an output clamped to (-1,1) using tanh. We train using mean squared error loss.

Deployment. At deployment, we query the classifier at each timestep with the observation and the base action. If it predicts an intervention, we query the residual policy and execute the combined base and residual actions; otherwise, we execute the base action alone. For the continual learning baselines HG-DAgger and EnsembleDAgger, we use real observations. However, for UPS (ours), we use decoded latent observations from the world model to enable us to imagine the outcomes of actions sampled from the combined policy. We generate $K/2 = 5$ action sequences using the combined policy and $K/2$ using the base policy alone to mitigate catastrophic forgetting in the event that the residual policy overcorrects toward modes underrepresented in the base policy.

Hyperparameter	Value
Batch Size	64
Learning Rate	1e-4
Iterations	10000
Dropout	0.2
Base Action Encoder Output Dimension	128
Action Predictor Hidden Dimension	256
Action Predictor Number of Hidden Layers	2

TABLE V: **Hyperparameters for Residual Policy Training.**

Role:
You are an expert Robotic Systems Monitor specializing in spatial reasoning, and failure detection.

Task:
Analyze the final outcome of a robot performing picking up the cup.

Visual Inputs: Each example contains a single image with two synchronized views:
- Left View (Left Pane): A global view of the workspace from the left side.
- Wrist View (Right Pane): A local, top-down view from the gripper.

Evaluation Rules (STRICT):
1. The image 4 is the final state of the task execution
2. Determine task outcome from the Wrist View and Front View:
 - GRASPING THE CUP IS A SUCCESS if you only see part of the cup round edge in the wrist view in the final frame.
 - GRASPING THE CUP IS A FAILURE if you can clearly see a complete round shape of the cup edge in the wrist view in the final frame.

Reference Examples (Ground Truth):
- Image 1: GRASPING THE CUP IS A SUCCESS because it grasps the cup from the center.
- Image 2: GRASPING THE CUP IS A FAILURE because the gripper is outside the cup.

Query: Analyze the Image 4 by comparing it against the reference examples (1-2) to tell which is the closest match.

Output Format :
1. One sentence only
2. tell the task outcome according to the rules
3. don't need to say the comparison between images

Fig. 15: **Hardware Prompt For Behavior Translation (Grasp)**

Role:
You are an expert Robotic Systems Monitor specializing in spatial reasoning, and failure detection.

Task:
Analyze the final outcome of a robot performing the task of picking up the cup and placing it in the bin.

Visual Inputs: Each example contains a single image with two synchronized views :
- Front View (Left Pane): A global view of the workspace. Use this view to determine the cup's location.
- Wrist View (Right Pane): A local, top-down view from the gripper. Use this view to evaluate whether the cup is placed in one of the bin or it is on the border.

Evaluation Rules (STRICT):
1. The image 3 is the final state of the task execution
2. Determine task outcome from the Wrist View:
  - SUCCESS if the cup is in either LEFT/RIGHT bin and standing upright.
  - FAILURE if the cup is on the border or it lies down in the bin.

Reference Examples (Ground Truth):
- Image 1: Successful because it places the cup in the LEFT bin and you can see the green cup is in the right part of the orange bin in the wrist view.
- Image 2: Successful because it places the cup in the RIGHT bin and you can see the green cup is in the left part of the orange bin in the wrist view.

Query:
Analyze the Image 3 by comparing it against the reference examples (1-2) to tell which is the closest match.

Output Format :
1. One sentence only
2. tell the placement of the cup [LEFT/RIGHT]
3. tell the task outcome
4. don't need to say the comparison between images

Fig. 16: **Hardware Prompt For Behavior Translation (Place)**

User Intent	Task Instruction
left	"Please place the nut on the peg with the handle facing left."
left	"I want the nut positioned on the peg with the handle directed left."
left	"Put the nut on the peg so that its handle is oriented leftward."
left	"Set the nut onto the peg with the handle pointing left."
left	"The nut goes on the peg with its handle turned to face left."
left	"Can the nut be placed on the peg so its handle is pointing to the left?"
left	"Set it up so that the nut's handle faces left while on the peg."
left	"The nut should be positioned on the peg with its handle going left."
left	"Place the nut on the peg such that its handle is directed to the left."
left	"I need the nut placed with a left-pointing handle on the peg."
left	"Pick up the nut, turn its handle to the left, and place it on the peg."
left	"Can you rotate the nut's handle to the left and set it on the peg?"
left	"The nut should go on the peg with its handle facing left."
left	"I want the nut placed on the peg so that I can easily grab it from the peg with my left hand."
left	"Please set the nut on the peg such that it is readily accessible using my left hand."
right	"The nut should go on the peg with its handle facing right."
right	"Make sure the nut's handle points right when you put it on the peg."
right	"I need you to place the nut so the handle is on the right side."
right	"Could you put the nut on the peg? The handle needs to face right."
right	"Place the nut on the peg such that its handle is directed to the right."
right	"Position the nut on the peg and turn the handle to face right."
right	"I need the nut placed with a right-pointing handle on the peg."
right	"Put the nut on the peg and make sure its handle faces right."
right	"The nut belongs on the peg with its handle turned rightward."
right	"When placing the nut on the peg, ensure its handle is oriented toward the right."
right	"With the handle pointing rightward, place the nut onto the peg."
right	"Turn the nut until its handle faces right, then set it on the peg."
right	"I'd like you to place the nut on the peg with its handle pointing to the right."
right	"Place the nut on the peg in a location my right hand can access with ease."
right	"Set the nut onto the peg so that right-hand retrieval is straightforward."
c.c.w. (left)	"Place the nut onto the peg with the handle directed at nine o'clock."
c.c.w. (left)	"I want the nut set on the peg so that its handle is pointing at nine o'clock."
c.c.w. (left)	"Can you please position the nut on the peg with its handle oriented toward nine o'clock."
c.c.w. (left)	"Set the nut onto the peg such that the handle faces the nine o'clock direction."
c.c.w. (left)	"The handle should be at nine o'clock when you set the nut on the peg."
c.c.w. (left)	"I need you to orient the nut's handle toward nine o'clock when you place it on the peg."
c.c.w. (left)	"When positioning the nut on the peg, make sure the handle faces nine o'clock please."
c.c.w. (left)	"The nut goes on the peg with its handle angled toward nine o'clock."
c.c.w. (left)	"Can you turn the nut counter-clockwise 90 degrees and set it on the peg?"
c.c.w. (left)	"Please rotate the nut a quarter turn counter-clockwise before putting it on the peg."
c.c.w. (left)	"I need you to give the nut a 90-degree counter-clockwise twist then place it on the peg."
c.c.w. (left)	"I want the nut rotated a quarter turn counter-clockwise prior to placing it on the peg."
c.c.w. (left)	"The nut should be turned a quarter turn counter-clockwise as you put it on the peg."
c.c.w. (left)	"Give the nut a 90-degree counter-clockwise rotation and then mount it on the peg."
c.c.w. (left)	"Before you set the nut on the peg, turn it a quarter turn counter-clockwise."
c.w. (right)	"When positioning the nut on the peg, make sure the handle faces three o'clock please."
c.w. (right)	"The nut goes on the peg with its handle angled toward three o'clock."
c.w. (right)	"I want you to place the nut on the peg while ensuring the handle is at three o'clock."
c.w. (right)	"Put the nut on the peg after rotating it so the handle points to three o'clock."
c.w. (right)	"With the handle at three o'clock, place the nut onto the peg."
c.w. (right)	"The nut should be placed on the peg with its handle pointing three o'clock."
c.w. (right)	"Set the nut on the peg, making sure the handle is directed at three o'clock."
c.w. (right)	"Please drop the nut onto the peg while keeping the handle oriented to three o'clock."
c.w. (right)	"Before you set the nut on the peg, turn it a quarter turn clockwise."
c.w. (right)	"Can you spin the nut 90 degrees clockwise while positioning it on the peg?"
c.w. (right)	"Turn the nut clockwise 90 degrees, then place it on the peg."
c.w. (right)	"I'd like you to rotate the nut a quarter turn clockwise and put it on the peg."
c.w. (right)	"Please give the nut a clockwise quarter turn before placing it onto the peg."
c.w. (right)	"Twist the nut clockwise by 90 degrees, then set it down on the peg."
c.w. (right)	"I need the nut turned a quarter turn clockwise when you place it on the peg."

TABLE VI: **Straightforward Simulation Task Instructions.** 60 straightforward task instructions used for the calibration and test datasets are listed here. 40 of these are randomly sampled to be part of the calibration set, while the remaining 20 make up the test set. We use "c.c.w." and "c.w." to denote "counterclockwise" and "clockwise" respectively.

User Intent	Task Instruction
left or right	“Please place the nut on the peg.”
left or right	“I want the nut positioned on the peg.”
left or right	“Pick up the nut and set it on the peg.”
left or right	“Can you set the nut on the peg?”
left or right	“Put the nut onto the peg.”
left or right	“I need you to place the nut on the peg.”
left or right	“Move the nut to the peg.”
left or right	“Could you position the nut on the peg?”
left or right	“Set the nut down on the peg.”
left or right	“I want you to put the nut on the peg.”
left or right	“Could you grab the nut and put it on the peg?”
left or right	“Please move the nut onto the peg.”
left or right	“The nut needs to go on the peg.”
left or right	“Can you mount the nut on the peg?”
left or right	“I’d like the nut placed onto the peg.”
left or right	“I’m going to need the nut positioned on the peg.”
left or right	“Would you mind putting the nut onto the peg?”
left or right	“Go ahead and place the nut on the peg.”
left or right	“Can you get the nut situated on the peg?”
left or right	“I’d like you to move the nut over to the peg.”
left or right	“Put that nut on the peg right now please.”
left or right	“Transfer the nut so it sits on the peg.”
left or right	“I’m asking you to get the nut onto the peg.”
left or right	“Would you be able to secure the nut on the peg?”
left or right	“Set that nut down on the peg for me please.”
left or right	“I need the nut transferred onto the peg.”
left or right	“Please get the nut mounted over the peg.”
left or right	“Could you move the nut over and place it on the peg?”
left or right	“I want that nut relocated to the peg.”
left or right	“Would you fix the nut onto the peg?”
left or right	“I want the nut placed on the peg so that its handle points toward my non-dominant hand side.”
left or right	“Put the nut on the peg with its handle turned in the direction of my non-preferred hand.”
left or right	“Could you move the nut onto the peg so that its handle is oriented toward my non-dominant hand?”
left or right	“Please set the nut on the peg with the handle facing the side of my non-preferred hand.”
left or right	“I need the nut positioned on the peg so that its handle points to my non-dominant hand side.”
left or right	“Place the nut on the peg with its handle directed toward my non-preferred hand.”
left or right	“Can you put the nut on the peg so that its handle is aligned with my non-dominant hand?”
left or right	“Set the nut onto the peg with its handle facing in the direction of my non-preferred hand.”
left or right	“I want you to place the nut on the peg so that its handle points toward my non-dominant hand side.”
left or right	“Could you position the nut on the peg with its handle turned toward my non-preferred hand?”
left or right	“Please put the nut on the peg so that its handle is oriented in line with my non-dominant hand.”
left or right	“I need you to set the nut on the peg with its handle facing toward my non-preferred hand side.”
left or right	“Mount the nut onto the peg so that its handle points in the direction of my non-dominant hand.”
left or right	“Can you place the nut on the peg with its handle aligned toward my non-preferred hand?”
left or right	“Set up the nut on the peg so that its handle is directed toward my non-dominant hand side.”
left or right	“Mount the nut onto the peg so that the handle is directed towards my dominant-hand side.”
left or right	“Can you please attach the nut onto the peg with the handle facing my preferred hand side?”
left or right	“I need the nut placed on the peg so that its handle points toward my dominant hand side.”
left or right	“Put the nut on the peg with its handle turned in the direction of my preferred hand side.”
left or right	“Please set the nut on the peg with the handle facing the side of my preferred hand.”
left or right	“I want the nut positioned on the peg so that its handle points to my dominant hand side.”
left or right	“Place the nut on the peg with its handle directed toward my preferred hand.”
left or right	“Can you put the nut on the peg so that its handle is aligned with my dominant hand?”
left or right	“Set the nut onto the peg with its handle facing in the direction of my preferred hand.”
left or right	“I want you to place the nut on the peg so that its handle points toward my dominant hand side.”
left or right	“Could you position the nut on the peg with its handle turned toward my preferred hand side?”
left or right	“Please put the nut on the peg so that its handle is oriented in line with my dominant hand.”
left or right	“I need you to set the nut on the peg with its handle facing toward my preferred hand side.”
left or right	“Mount the nut onto the peg so that its handle points in the direction of my dominant hand.”
left or right	“Can you place the nut on the peg with its handle aligned toward my preferred hand?”

TABLE VII: **Ambiguous Simulation Task Instructions.** 60 ambiguous task instructions used for the calibration and test datasets are listed here. 40 of these are randomly sampled to be part of the calibration set, while the remaining 20 make up the test set.

Role:
You are an expert Robotic Systems Monitor specializing in spatial reasoning, and failure detection.

Task:
Based on the user's instruction and the possible outcomes of different behavior modes provided, your job is to select the behavior mode whose outcome that best fulfills the user's instruction.

Evaluation Phase: {QUERY} Phase.

{NOTE}

Evaluation Rules (STRICT) :
1. Evaluate the options based on the evaluation phase and the user's instruction. If it is grasping phase, just selects for appropriate grasping.
2. If no mode fully meets the user's intent, you must choose "an option not listed here".
3. Assume that each mode's outcome is exactly as described in the options, with no adjustments.

Response format:
1. Begin your response with the single letter corresponding to your choice.
2. Repeat the description of that mode.
3. Provide a one-sentence reason explaining why it is the most appropriate choice.

User instruction:
{USER_INSTRUCTION}

Fig. 17: **Hardware Prompt for VLM Verification in Forewarn**

Environment setup: You are a robot arm situated on one side of a tabletop and a user stands on the other side. All directions referenced should be from the user's perspective.

Goal: Given a set of objects on the table, and a general task goal, your job is to generate a set of possible user intents that would describe different ways the user might want to complete the task at the given phase.

Clarifications: If the task goal involves placing an object, please focus on different positions for the object at the end of the task.

Objects on table: two coffee pads and two coffee machines
Phase: grasp
Environment: the table has one coffee pod and one coffee machine (on on top and one on bottom).
Task goal: the goal is to put the coffee pod in the machine
Possible user intents:
A) pick up the coffee pod.

Objects on table: one apple and two knives
Phase: cut
Environment: the table has one apple and two knives, one is large and the other is small.
Task goal: the goal is to cut the apple with the knife
Possible user intents:
A) cut the apple with the large knife.
B) cut the apple with the small knife.

Objects on table: a cup and two bins
Phase: [QUERY]
Environment: the table has two bins (left bin and right bin)
Task goal: the goal is to place the cup in the bin.
Possible user intents:

Fig. 18: **Hardware Prompt for Generating Possible Intents**

Role:
You are an expert Robotic Systems Monitor specializing in spatial reasoning, and failure detection to guide a robot to place the cup in the bin.

Task:
Based on the instruction that the user provides to the robot, your job is to select the intent that best matches the user's instruction. Assign each possible intent a likelihood score based on how probable it is given the instruction that the user provides to the robot.
1.0 means the given intent is the only feasible option given the user's instruction, and 0.0 means the given intent is completely at odds with the user's instruction.
Ensure that the scores sum to 1.0 across all modes.
If multiple options seem equally likely, please distribute the likelihood scores evenly among those options.

Clarifications:
If the user's instruction is underspecified, please do not make any biased assumptions about the user's intent or preferences.
There is one cup and two orange bins on the table.
[NOTE]

<self-score version>
Format your response as follows:
1. If the Intent Options has only A, output is: {"A": <score for option A>}
2. If the Intent Options has A and B, output is: {"A": <score for option A>, "B": <score for option B>}
3. If the Intent Options has A , B, C, output is: {"A": <score for option A>, "B": <score for option B>,"C": <score for option C>}
4. If the Intent Options has A, B, C, D, output is: {"A": <score for option A>, "B": <score for option B>, "C": <score for option C>, "D": <score for option D>}
Do not include any other text in your response, just the JSON object with the scores.

<logit version>
Format your response as follows:
Provide only the single letter that corresponds to the selected option.

User Instruction:
[USER_INSTRUCTION]

Intent Options:
[INTENT]

Fig. 19: **Hardware Prompt for Inferring Intent Probability**

You are a helpful assistant that reformulates ambiguous instructions into clear, specific ones.

Original instruction from user:
"{original_instruction}"

The available options were:
{options}

To clarify the ambiguity, the following question was asked:
"{clarification_question}"

The human answered:
"{human_answer}"

Your task is to rewrite the original instruction to be more specific and unambiguous, incorporating the human's answer. The updated instruction should:
1. Preserve the intent of the original instruction
2. Clearly specify which option was chosen based on the human's answer
3. Be concise and actionable
4. Sound natural (as if the user had given this instruction from the start)

Generate ONLY the updated instruction, nothing else. Do not include any preamble or explanation.

Fig. 20: **Hardware Prompt for Updating Instructions**

You are a helpful robot assistant that needs to clarify ambiguous instructions before executing tasks.

The user has given you the following instruction:
"{user_instruction}"

However, this instruction is ambiguous because there are multiple possible options:
{options}

Your task is to generate a single, clear, and concise clarification question to ask the human user. The question should:
1. Be polite and natural-sounding
2. Clearly present the available options
3. Help resolve the ambiguity so you can complete the task correctly

Generate ONLY the clarification question, nothing else. Do not include any preamble or explanation.

Fig. 21: **Hardware Prompt for Clarification Questions**

Role:
You are an expert Robotic Systems Monitor specializing in spatial reasoning, and failure detection.

Task:
Based on the user's instruction and the possible outcomes of different behavior modes provided, your job is to select the behavior mode whose outcome that best fulfills the user's instruction.

Rules:
1.Assign each possible behavior mode a likelihood score based on how probable it is given the instruction that the user provides to the robot.
2. 1.0 means the given behavior mode is the only feasible option given the user's instruction, and 0.0 means the given behavior mode is completely at odds with the user's instruction.
3. Ensure that the scores sum to 1.0 across all modes and this is very important!!!
4. If multiple options seem equally likely, please distribute the likelihood scores evenly among those options.
5. You cannot give all 0.0 for all the options available.
6. You can give any value between 0 and 1 and must give a probability value representing your own uncertainty rather than exactly 0 or 1!! Don't use 0 or 1!

Evaluation Phase: [QUERY] Phase.

[NOTE]

Evaluation Rules (STRICT):
1. Evaluate the options based on the evaluation phase and the user's intent provided to the robot. If it is grasping phase, just selects for appropriate grasping.
2. If no mode fully meets the user's intent, you must choose "an option not listed here" over the failures.
3. Assume that each mode's outcome is exactly as described in the options, with no adjustments.

<self-score version>
Format your response as follows:
{"A": <score for option A>, "B": <score for option B>, "C": <score for option C>, "D": <score for option D>}
Do not include any other text in your response, just the JSON object with the scores. Only include available options in the json and follow the rules.

<logit version>
Format your response as follows:
Provide only the single letter that corresponds to the selected option.

User Intent: [USER_INSTRUCTION]

Multiple Choice Options:
[MCQ_OPTIONS]

Fig. 22: **Hardware Prompt for Generating Behavior Probability Conditioned On the Intent**

Role:
You are a Robotic Systems Monitor to monitor the process of placing a cup in the bin. Your task is to determine if a user instruction contains enough discriminative information to select a single option from the provided options.

Task: Evaluate the user instruction against the options.

Step 1 goal: Analyze whether the instruction successfully narrows the choice to exactly one behavior mode.

Environment Information:

There is only one cup and two bins on the table.
[NOTE]

Logic for Evaluation:

Is it Determinate? Based on the environment information, if the instruction contains a specific modifier (location, color, state, tag, or label) that maps clearly to one of the options, there is no ambiguity.

Is it Indeterminate? If the instruction is generic (e.g., "put it away," "in a bin") or uses a descriptor that could apply to multiple options (e.g., "the bin" when two bins exist), then there is ambiguity.

No Over-thinking: Do not invent secondary meanings beyond the environment information. Don't think about some ambiguity in the process and we only consider the outcome. There is no multiple cups. If a user specifies a side or a category, assume they are being literal and specific.

Format:

Ambiguity: [Yes/No]

Reason: [One concise sentence explaining whether a specific identifier was provided or if the language was too generic.]

User Instruction: [USER_INSTRUCTION]

Multiple Choice Options:
[MCQ_OPTIONS]

Fig. 23: **Hardware Prompt For Asking VLM to Analyze Ambiguity First in CoT Reasoning**

Role:
You are a Robotic Systems Monitor to monitor the process of placing a cup in the bin. Your task is to determine if a user instruction contains enough discriminative information to select a single option from the provided options.

Task: Evaluate the user instruction against the options.

User Instruction: [USER_INSTRUCTION]

Multiple Choice Options:
[MCQ_OPTIONS]

Reasoning: [STEP1_REASON]

Evaluation Phase: [QUERY] Phase.

Step 2 goal:
Based on the instruction that the user provides to the robot and your prior reasoning, your job is to select the behavior mode that best fulfills the user's instruction.

Rules:
1. Evaluate the options based on the evaluation phase, the reasoning and user instruction provided to the robot. If the evaluation phase is grasping, just selects the behavior mode for good grasping that can move to the next phase.
2. Assign each possible behavior mode a likelihood score based on how probable it is given the instruction that the user provides to the robot.
3. 1.0 means the given behavior mode is the only feasible option given the user's instruction, and 0.0 means the given behavior mode is completely at odds with the user's instruction.
4. Ensure that the scores sum to 1.0 across all modes and this is very important!!!
5. If multiple options seem equally likely, please distribute the likelihood scores evenly among those options.
6. You cannot give all 0.0 for all the options available.
7. You can give any value between 0 and 1 and must give a probability value representing your own uncertainty rather than exactly 0 or 1!! Don't use 0 or 1!

Clarifications:
If the user's instruction is underspecified, please do not make any biased assumptions about the user's intent or preferences.

<self-score version>
Format your response for step 1 as follows:
{"A": <score for option A>, "B": <score for option B>, "C": <score for option C>, "D": <score for option D>}
Do not include any other text in your response, just the JSON object with the scores. Only include available options in the json and follow the rules.

<logit version>
Format your response as follows:
Provide only the single letter that corresponds to the selected option.

Fig. 24: **Hardware Prompt For Asking VLM to Score Options Based On Previous Reasoning in CoT Reasoning**

Role:
You are an expert Robotic Systems Monitor specializing in spatial reasoning, and failure detection.

Task:
Based on the user's instruction and the possible outcomes of different behavior modes provided, your job is to select the behavior mode whose outcome that best fulfills the user's instruction.
Assign each option a likelihood score based on how probable it is given the instruction that the user provides to the robot.
1.0 means this is only feasible option given the user's instruction, and 0.0 means the option is completely at odds with the user's instruction.
Ensure that the scores sum to 1.0 across all modes.
If multiple options seem equally likely, please distribute the likelihood scores evenly among those options.

Evaluation Phase: [QUERY] Phase.

[NOTE]

Clarifications:
If the user's instruction is underspecified, please do not make any biased assumptions about the user's intent or preferences.

Evaluation Rules (STRICT):
1. Evaluate the options based on the evaluation phase and the user's instruction. If it is grasping phase, just selects for appropriate grasping.
2. If no mode fully satisfies the user's instruction, you must choose "an option not listed here" rather than other options.
3. Assume that each mode's outcome is exactly as described in the options, with no adjustments.

<self-score version>
Response format:
{"A": <score for option A>, "B": <score for option B>, "C": <score for option C>, "D": <score for option D>}
Do not include any other text in your response, just the JSON object with the scores.

<logit version>
Format your response as follows:
Provide only the single letter that corresponds to the selected option.

User instruction:
[USER_INSTRUCTION]

Multiple Choice Options:
[MCQ_OPTIONS]

Fig. 25: **Hardware Prompt For Directly Generating Scores for Available Options**

Role:
You are an expert Robotic Systems Monitor specializing in spatial reasoning, assembly verification, and failure detection.

Task:
Analyze the final outcome of a robot performing a square-nut peg-in-hole assembly from the last frame of the rollout.

Visual Inputs:
Each example contains two synchronized views:
- Front View (Left Pane): A global view of the workspace. Use this view to determine if the robot successfully grasps the nut and the nut moves with the robot gripper.
- Wrist View (Right Pane): A local, top-down view from the gripper. Use this view to evaluate if the handle is between the robot grippers.

Evaluation Rules (STRICT):
1. The Image 4 provided is the final state of the task execution
2. Determine task outcome from the Wrist View and Front View:
  - GRASPING THE NUT IS A SUCCESS if the gripper grasps the nut and the handle of the nut is between the grippers in the wrist view and the hole of the nut is centered in the wrist view.
  - GRASPING THE NUT IS A FAILURE if any of the following is true: 1. the gripper doesn't grasp the nut, 2. the nut is tilted in the wrist view, or 3. the handle is not centered and held securely between the grippers in the wrist view.

Reference Examples (Ground Truth):
- Image 1: GRASPING THE NUT IS A SUCCESS because it grasps the handle of the nut tightly in its grippers and the hole of the nut is centered in its wrist view.
- Image 2: GRASPING THE NUT IS A FAILURE because the nut is not centered between the grippers in the wrist view.

Query:
Analyze Image 4 using the evaluation rules by comparing it against the reference examples (1-2) to tell which is the closest match.

Output Format:
1. One sentence only
2. Describe the task outcome according to the rules
3. Don't need to explicitly say the comparison between images

Fig. 26: **Simulation Prompt For Behavior Translation (Grasp)**

Role:
You are an expert Robotic Systems Monitor specializing in spatial reasoning, assembly verification, and failure detection.

Task:
Analyze the final outcome of a robot performing a square-nut peg-in-hole assembly from the last frame of the rollout.

Visual Inputs:
Each example contains two synchronized views:
- Front View (Left Pane): A global view of the workspace. Use this view to determine which direction the handle of the nut is facing.
- Wrist View (Right Pane): A local, top-down view from the gripper. Use this view to evaluate if hole of the nut is properly aligned with the peg.

Evaluation Rules (STRICT):
1. Image 4 provided is the final state of the task execution
2. Determine task outcome from the Wrist View and Front View:
- ALIGNING THE NUT IS A SUCCESS HANDLE-FACING-LEFT if both of the following are true: 1. the square hole of the nut is aligned with the square peg (i.e., the peg lies roughly within the inner borders of the nut's hole) in the wrist view and 2. the end effector of the robot is to the left of the peg in the middle of the image in the front view.
- ALIGNING THE NUT IS A SUCCESS HANDLE-FACING-RIGHT if both of the following are true: 1. the square hole of the nut is aligned with the square peg (i.e., the peg lies roughly within the inner borders of the nut's hole) in the wrist view and 2. the end effector of the robot is to the right of the peg on the far right of the image in the front view.
- ALIGNING THE NUT IS A FAILURE if any of the following is true: 1. the nut is not positioned at the top of the peg in the front view, 2. the tan peg does not appear inside the hole of the nut in the wrist view (i.e. the interior of the hole of the nut is all white), or 3. the square hole of the nut is significantly misaligned with the square peg.

Reference Examples (Ground Truth):
- Image 1: ALIGNING THE NUT IS A SUCCESS HANDLE-FACING-LEFT because the hole of the nut is aligned with the peg in the wrist view and the end effector of the robot is to the left of the peg in the front view.
- Image 2: ALIGNING THE NUT IS A SUCCESS HANDLE-FACING-RIGHT because the hole of the nut is aligned with the peg in the wrist view and the end effector of the robot is to the right of the peg in the front view.
- Image 3: ALIGNING THE NUT IS A FAILURE because in the front view, the nut lies on the table, well before reaching the peg, and in the wrist view, the interior of the square hole of the peg is completely white and empty.

Query:
Analyze Image 4 using the evaluation rules by comparing it against the reference examples (1-3) to tell which is the closest match.

Output Format:
1. One sentence only
2. Describe the task outcome and direction of the nut (SUCCESS or FAILURE and, if SUCCESS, HANDLE-FACING-RIGHT or HANDLE-FACING-LEFT) according to the rules
3. Don't need to explicitly say the comparison between images

Fig. 27: **Simulation Prompt For Behavior Translation (Place)**

Task setup:
You are an expert Robotic Systems Monitor specializing in spatial reasoning and failure detection.
You are guiding a robot to complete a nut assembly task.
The robot is situated on one side of a table and the user stands on the other side. All directions referenced are from the user's perspective.
The nut begins with its handle facing backwards, away from the user, on the table.
All directions, rotations, and references in the behavior modes are from the user's perspective.

Goal:
Based on the instruction that the user provides to the robot, your job is to select the behavior mode that best fulfills the user's instruction.

Clarifications:
1. If the user's instruction is underspecified, please do not make any biased assumptions about the user's intent or preferences.
2. Likewise, do not make assumptions about whether the user is left or right handed; assume both handednesses are equally likely.
3. Note that because directions and rotations are from the user's perspective, rotating the nut clockwise ninety degrees will result in the handle facing right and rotating it counterclockwise ninety degrees will result in the handle facing left.
4. Likewise, when it faces the three o'clock position, it is facing right and when it faces the nine o'clock position, it is facing left.
5. Do not focus on the intermediate motion of the nut, instead focus on the end outcome and whether this achieves the user's instruction.

Evaluation Phase: [QUERY] Phase.

Evaluation Rules (STRICT):
1. Evaluate the options based on the evaluation phase and the user's instruction provided to the robot. If it is grasping phase, just select the mode for appropriate grasping.
2. If no mode fully meets the user's intent, you must choose "an option not listed here" over any failures.
3. Assume that each mode's outcome is exactly as described in the options, with no adjustments.

Response format:
- Begin your response with the single letter corresponding to your choice.
- Repeat the description of that mode.
- Provide a one-sentence reason explaining why it is the most appropriate choice.

User instruction:
[USER_INSTRUCTION]

Fig. 28: **Simulation Prompt for VLM Verification in Forewarn**

Environment setup: You are a robot arm situated on one side of a tabletop and a user stands on the other side. All directions referenced (left, right, etc.) must be from the user's perspective.

Goal: Given a set of objects on the table and a general task goal, generate a list of distinct, valid user intents that resolve ambiguity in how the task is completed.

Clarifications:
- Grasp Phase: There is no ambiguity in the grasp. Generate exactly one intent in the grasp phase. Generate intents only for your current phase.
- If the task involves placing an object into a specific receptacle or onto a fixture (like a peg or slot), the ambiguity is often in the object's **orientation**.
- Focus specifically on the direction of functional parts (like handles) relative to the user (e.g., facing left vs. facing right).
- Do not list orientations that are physically impossible or unlikely for the grasp (e.g., handle facing away or towards the user).

Objects on table: two coffee pads and two coffee machines
Phase: grasp
Environment: the table has one coffee pod and one coffee machine (one on top and one on bottom).
Task goal: the goal is to put the coffee pod in the machine
Possible user intents:
A) pick up the coffee pod.

Objects on table: one mug and one shelf
Phase: grasp
Environment: the table has one mug with a handle and one shelf.
Task goal: the goal is to put the mug on the shelf
Possible user intents:
A) pick up the mug by the handle

Objects on table: one mug and one shelf
Phase: place
Environment: the table has one mug with a handle and one shelf.
Task goal: the goal is to put the mug on the shelf
Possible user intents:
A) place the mug on the shelf with the handle facing left.
B) place the mug on the shelf with the handle facing right.

Objects on table: a square nut with a handle and a square peg
Phase: [QUERY]
Environment: the peg is pointing upwards. The nut has a handle protruding from one side.
Task goal: the goal is to place the nut on the peg
Possible user intents:

Fig. 29: **Simulation Prompt for Generating Possible Intents**

Task setup:
You are an expert Robotic Systems Monitor specializing in spatial reasoning and failure detection.
You are guiding a robot to complete a nut assembly task.
The robot is situated on one side of a table and the user stands on the other side. All directions referenced are from the user's perspective.
The nut begins with its handle facing backwards, away from the user, on the table.
All directions, rotations, and references in the behavior modes are from the user's perspective.

<self-score version>
Goal:
Assign each possible intent a likelihood score based on how probable it is given the instruction that the user provides to the robot.
1.0 means the given intent is the only feasible option given the user's instruction, and 0.0 means the given intent is completely at odds with the user's instruction.
Ensure that the scores sum to 1.0 across all modes.
If multiple options seem equally likely, please distribute the likelihood scores evenly among those options.

<logit version>
Goal:
Based on the instruction that the user provides to the robot, your job is to select the intent that best matches the user's instruction.

Clarifications:
1. If the user's instruction is underspecified, please do not make any biased assumptions about the user's intent or preferences.
2. Likewise, do not make assumptions about whether the user is left or right handed; assume both handednesses are equally likely.
3. Note that because directions and rotations are from the user's perspective, rotating the nut clockwise ninety degrees will result in the handle facing right and rotating it counterclockwise ninety degrees will result in the handle facing left.
4. Likewise, when it faces the three o'clock position, it is facing right and when it faces the nine o'clock position, it is facing left.
5. Do not focus on the intermediate motion of the nut, instead focus on the end outcome and whether this achieves the user's instruction.

<self-score version>
Format your response as follows:
{"A": <score for option A>, "B": <score for option B>, "C": <score for option C>, "D": <score for option D>}
Only include the options that are present in the multiple choice provided.
Do not include any other text in your response, just the JSON object with the scores.

<logit version>
Format your response as follows:
Provide only the single letter that corresponds to the selected option.

User Instruction: [USER_INSTRUCTION]

Multiple Choice Options:
[MCQ_OPTIONS]

Fig. 30: **Simulation Prompt for Inferring Intent Probability**

Task setup:
You are an expert Robotic Systems Monitor specializing in spatial reasoning and failure detection.
You are guiding a robot to complete a nut assembly task.
The robot is situated on one side of a table and the user stands on the other side. All directions referenced are from the user's perspective.
The nut begins with its handle facing backwards, away from the user, on the table.
All directions, rotations, and references in the behavior modes are from the user's perspective.

Step 2 goal:
Based on the instruction that the user provides to the robot and your prior reasoning, your job is to select the behavior mode that best fulfills the user's instruction.

<self-score version>
Rules:
1. Assign each behavior mode a likelihood score based on how probable it is given the instruction that the user provides to the robot.
2. 1.0 means the given mode is the only feasible option to satisfy the user's instruction, and 0.0 means the given mode is completely at odds with the user's instruction.
3. Ensure that the scores sum to 1.0 across all modes.
4. If multiple options seem equally likely, distribute the likelihood scores evenly among those options.
5. You cannot give all 0.0 for all the options available.
6. You can give any value between 0.0 and 1.0 and must give a probability value representing your own uncertainty rather than exactly 0.0 or 1.0. DO NOT use exactly 0.0 or 1.0!

Clarifications:
1. If the user's instruction is underspecified, please do not make any biased assumptions about the user's intent or preferences.
2. Likewise, do not make assumptions about whether the user is left or right handed; assume both handednesses are equally likely.
3. Note that because directions and rotations are from the user's perspective, rotating the nut clockwise ninety degrees will result in the handle facing right and rotating it counterclockwise ninety degrees will result in the handle facing left.
4. Likewise, when it faces the three o'clock position, it is facing right and when it faces the nine o'clock position, it is facing left.
5. Do not focus on the intermediate motion of the nut, instead focus on the end outcome and whether this matches the user's instruction.

Evaluation Phase: [QUERY] Phase.

<self-score version>
Format your response for step 2 as follows:
{"A": <score for option A>, "B": <score for option B>, "C": <score for option C>, "D": <score for option D>}
Only include the options that are present in the multiple choice provided.
Do not include any other text in your response, just the JSON object with the scores.

<logit version>
Format your response for step 2 as follows:
Provide only the single letter that corresponds to the selected option.

User Instruction: [USER_INSTRUCTION]

Multiple Choice Options:
[MCQ_OPTIONS]

Reasoning: [STEP1_RESPONSE]

Fig. 31: **Simulation Prompt For Asking VLM to Analyze Ambiguity First in CoT Reasoning**

Task setup:
You are an expert Robotic Systems Monitor specializing in spatial reasoning and failure detection.
You are guiding a robot to complete a nut assembly task.
The robot is situated on one side of a table and the user stands on the other side. All directions referenced are from the user's perspective.
The nut begins with its handle facing backwards, away from the user, on the table.
All directions, rotations, and references in the behavior modes are from the user's perspective.

Step 1 goal:
Based on the instruction that the user provides to the robot and the available behavior modes, analyze whether the instruction successfully narrows the choice to exactly one behavior mode.

Clarifications:
1. If the user's instruction is underspecified, please do not make any biased assumptions about the user's intent or preferences.
2. Likewise, do not make assumptions about whether the user is left or right handed; assume both handednesses are equally likely.
3. Note that because directions and rotations are from the user's perspective, rotating the nut clockwise ninety degrees will result in the handle facing right and rotating it counterclockwise ninety degrees will result in the handle facing left.
4. Likewise, when it faces the three o'clock position, it is facing right and when it faces the nine o'clock position, it is facing left.
5. Do not focus on the intermediate motion of the nut, instead focus on the end outcome and whether this matches the user's instruction.

Evaluation Phase: [QUERY] Phase.

Format your response for step 1 as follows:
Begin by saying "Yes, there is ambiguity" (if there are multiple possible conditions) or "No, there is not ambiguity" if there is only one possible condition.
Follow by providing a concise, one-sentence justification for your answer.

User Instruction: [USER_INSTRUCTION]

Multiple Choice Options:
[MCQ_OPTIONS]

Fig. 32: **Simulation Prompt For Asking VLM to Score Options Based On Previous Reasoning in CoT Reasoning**

Task setup:
You are an expert Robotic Systems Monitor specializing in spatial reasoning and failure detection.
You are guiding a robot to complete a nut assembly task.
The robot is situated on one side of a table and the user stands on the other side. All directions referenced are from the user's perspective.
The nut begins with its handle facing backwards, away from the user, on the table.
All directions, rotations, and references in the behavior modes are from the user's perspective.

Goal:
Based on the intent that the user provides to the robot, your job is to select the behavior mode that best fulfills the user's intent.

<self-score version>
Rules:
1. Assign each behavior mode a likelihood score based on how probable it is given the instruction that the user provides to the robot.
2. 1.0 means the given mode is the only feasible option to satisfy the user's instruction, and 0.0 means the given mode is completely at odds with the user's instruction.
3. Ensure that the scores sum to 1.0 across all modes.
4. If multiple options seem equally likely, distribute the likelihood scores evenly among those options.
5. You cannot give all 0.0 for all the options available.
6. You can give any value between 0.0 and 1.0 and must give a probability value representing your own uncertainty rather than exactly 0.0 or 1.0. DO NOT use exactly 0.0 or 1.0!

Evaluation Phase: [QUERY] Phase.

Evaluation Rules (STRICT):
1. Evaluate the options based on the evaluation phase and the user's instruction provided to the robot. If it is grasping phase, just select the mode for appropriate grasping.
2. If no mode fully meets the user's intent, you must choose "an option not listed here" over any failures.
3. Assume that each mode's outcome is exactly as described in the options, with no adjustments.

<self-score version>
Format your response as follows:
{"A": <score for option A>, "B": <score for option B>, "C": <score for option C>, "D": <score for option D>}
Only include the options that are present in the multiple choice provided.
Do not include any other text in your response, just the JSON object with the scores.

<logit version>
Format your response as follows:
Provide only the single letter that corresponds to the selected option.

User Intent: [USER_INSTRUCTION]

Multiple Choice Options:
[MCQ_OPTIONS]

Fig. 33: **Simulation Prompt for Generating Behavior Probability Conditioned On the Intent**

Task setup:
You are an expert Robotic Systems Monitor specializing in spatial reasoning and failure detection.
You are guiding a robot to complete a nut assembly task.
The robot is situated on one side of a table and the user stands on the other side. All directions referenced are from the user's perspective.
The nut begins with its handle facing backwards, away from the user, on the table.
All directions, rotations, and references in the behavior modes are from the user's perspective.

Goal:
Based on the instruction that the user provides to the robot, your job is to select the behavior mode that best fulfills the user's instruction.

<self-score version>
Rules:
1. Assign each behavior mode a likelihood score based on how probable it is given the instruction that the user provides to the robot.
2. 1.0 means the given mode is the only feasible option to satisfy the user's instruction, and 0.0 means the given mode is completely at odds with the user's instruction.
3. Ensure that the scores sum to 1.0 across all modes.
4. If multiple options seem equally likely, distribute the likelihood scores evenly among those options.
5. You cannot give all 0.0 for all the options available.
6. You can give any value between 0.0 and 1.0 and must give a probability value representing your own uncertainty rather than exactly 0.0 or 1.0. DO NOT use exactly 0.0 or 1.0!

Clarifications:
1. If the user's instruction is underspecified, please do not make any biased assumptions about the user's intent or preferences.
2. Likewise, do not make assumptions about whether the user is left or right handed; assume both handednesses are equally likely.
3. Note that because directions and rotations are from the user's perspective, rotating the nut clockwise ninety degrees will result in the handle facing right and rotating it counterclockwise ninety degrees will result in the handle facing left.
4. Likewise, when it faces the three o'clock position, it is facing right and when it faces the nine o'clock position, it is facing left.
5. Do not focus on the intermediate motion of the nut, instead focus on the end outcome and whether this achieves the user's instruction.

Evaluation Phase: [QUERY] Phase.

Evaluation Rules (STRICT):
1. Evaluate the options based on the evaluation phase and the user's instruction provided to the robot. If it is grasping phase, just select the mode for appropriate grasping.
2. If no mode fully meets the user's intent, you must choose "an option not listed here" over any failures.
3. Assume that each mode's outcome is exactly as described in the options, with no adjustments.

<self-score version>
Format your response as follows:
{"A": <score for option A>, "B": <score for option B>, "C": <score for option C>, "D": <score for option D>}
Only include the options that are present in the multiple choice provided.
Do not include any other text in your response, just the JSON object with the scores.

<logit version>
Format your response as follows:
Provide only the single letter that corresponds to the selected option.

User Instruction: [USER_INSTRUCTION]

Multiple Choice Options:
[MCQ_OPTIONS]

Fig. 34: **Simulation Prompt For Directly Generating Scores for Available Options**