Generalizable Dense Reward for Long-Horizon Robotic Tasks

Silong Yong[1], Stephen Sheng[2], Carl Qi[3], Xiaojie Wang[2]
Evan Sheehan[2], Anurag Shivaprasad[2], Yaqi Xie[1]
Katia Sycara[1], Yesh Dattatreya[2]

[1]Carnegie Mellon University [2]Amazon Robotics [3]UT Austin

***Abstract*— Existing robotic foundation policies are trained primarily via large-scale imitation learning. While such models demonstrate strong capabilities, they often struggle with long-horizon tasks due to distribution shift and error accumulation. While reinforcement learning (RL) can finetune these models, it cannot work well across diverse tasks without manual reward engineering. We propose VLLR, a dense reward framework combining (1) an extrinsic reward from Large Language Models (LLMs) and Vision-Language Models (VLMs) for task progress recognition, and (2) an instrinsic reward based on policy self-certainty. VLLR uses LLMs to decompose tasks into verifiable subtasks and then VLMs to estimate progress to initialize the value function for a brief warm-up phase, avoiding prohibitive inference cost during full training; and self-certainty provides per-step intrinsic guidance throughout PPO finetuning. Ablation studies reveal complementary benefits: VLM-based value initialization primarily improves task completion efficiency, while self-certainty primarily enhances success rates, particularly on out-of-distribution tasks. On the CHORES benchmark covering mobile manipulation and navigation, VLLR achieves up to 56% absolute success rate gains over the pretrained policy, up to 5% gains over state-of-the-art RL finetuning methods on in-distribution tasks, and up to 10% gains on out-of-distribution tasks, all without manual reward engineering. Additional visualizations can be found here.**

I. INTRODUCTION

Long-horizon robotic tasks require a robot to compose multiple skills before receiving meaningful supervision. In mobile manipulation [1], [2], for example, a robot may need to navigate across rooms, search for an object, reposition obstacles, and finally grasp the target. While recent foundation policies trained via large-scale imitation learning [3], [1] demonstrate strong individual skills, they often struggle when these skills must be composed over extended horizons. Errors accumulate, and sparse end-of-task success signals provide little guidance for correcting intermediate mistakes.

We study how to effectively finetune pretrained foundation policies for long-horizon tasks using reinforcement learning. Although RL provides a natural mechanism for policy improvement through interaction [2], [4], [5], [6], its effectiveness critically depends on reward design [7]. In long-horizon robotics, relying solely on sparse success signals leads to severe credit assignment problems, while manually engineered dense rewards are brittle and task-specific. Moreover, reward signals must remain grounded in perception and physical interaction, making scalable reward modeling fundamentally challenging.

We argue that reward design for foundation policy finetuning in long-horizon settings must satisfy two requirements: (1) provide semantic supervision at the level of subgoals to alleviate sparse credit assignment; and (2) provide dense per-step feedback to guide local action refinement.

To satisfy the first requirement, we reconceptualize reward modeling in long-horizon robotics as a problem of *exposing and leveraging latent task hierarchy*. Rather than relying on sparse terminal supervision or manually engineered shaping signals, we explicitly model the semantic structure that underlies compositional tasks. Long-horizon behavior is inherently hierarchical: successful execution emerges from completing a sequence of intermediate subgoals that progressively approach task goals.

Instead of manually defining subgoals, we use a large language model (LLM) to extract intermediate objectives from high-level task descriptions. Crucially, this decomposition is conditioned on scene-graph representations of the environment, enabling globally consistent and perception-aware planning rather than purely textual reasoning.

We then turn the subgoals into explicit intermediate objectives that provide structured supervision during reinforcement learning. We employ a vision-language model (VLM) to evaluate the alignment between the agent's observation and each subgoal, yielding a progress estimate that reflects task advancement, which is subsequently turned into a reward that encourages behaviors that complete the subgoals.

To satisfy the second requirement, we introduce *policy self-certainty* [8], i.e. a metric that quantifies the confidence of the action distribution, as an intrinsic reward signal. We conceptualize long-horizon control as an implicit reasoning process embedded within pretrained foundation policies. Analogous to long-chain reasoning in large language models, sequential decision-making requires maintaining coherent internal representations over extended horizons. Recent work shows that self-certainty strongly correlates with reasoning correctness in language models [8]. Inspired by prior work that justifies the existence of an implicit world model inside a general policy [9], we propose the following insight: *self-certainty serves as a proxy for whether the policy's latent world model is internally consistent and progressing toward task completion*. This establishes a principled mechanism for leveraging knowledge already encoded within foundation policies, rather than relying solely on manually engineered external rewards. To prevent overconfidence in erroneous behaviors, we combine this intrinsic signal with the sparse task-success reward during fine-tuning. In practice, the task-success reward is assigned a larger weight in the overall

reward function, ensuring that the optimization objective remains anchored to the true task goal.

Building on these principles, we introduce Vision-Language-Long-horizon Reward (VLLR), a structured reward framework for fine-tuning foundation policies in long-horizon robotic tasks. VLLR combines grounded subgoal decomposition and intrinsic self-certainty to enable scalable and generalizable policy refinement.

We evaluate VLLR on the CHORES benchmark [1], which features compositional mobile manipulation and navigation tasks. On in-distribution tasks, VLLR improves absolute success rates by up to 56% over the pretrained foundation policy and consistently outperforms state-of-the-art RL fine-tuning methods based solely on sparse rewards. Importantly, VLLR generalizes to out-of-distribution task compositions, achieving up to 10% higher success rates than prior methods. These results demonstrate that structured semantic distillation combined with intrinsic progress modeling provides a practical pathway for extending foundation model fine-tuning to long-horizon robotic control.

In summary, we make the following contributions:

- We propose two key principles for reward modeling in long-horizon robotic fine-tuning which guide the design of our reward model: (i) leveraging task hierarchy to provide subgoal-level supervision, and (ii) exploiting intrinsic policy signals to obtain dense per-state feedback.
- We propose **Vision-Language Long-horizon Reward (VLLR)**, which uses VLM-derived semantic progress for *value initialization* and policy self-certainty as an intrinsic reward for stable long-horizon fine-tuning.
- On CHORES, VLLR, without manual engineering, achieves up to 56% absolute success rate gains over the pretrained policy and consistently outperforms state-of-the-art RL fine-tuning methods using sparse rewards.

II. Related Work

A. Robotic Foundation Models

Inspired by recent success in developing foundation models in language [10] and vision [11], prior work has developed foundation models that unify language, vision and action in robotics [3], [12], [13], [1], [14], [15], [16]. They share a similar training recipe that relies on large scale data and imitation learning, which produces generalizable policy models across diverse tasks. However, they still face challenges when dealing with long-horizon complex scenarios [17]. Our work addresses this issue by developing finetuning signal for better generalizability of these policies.

B. Reinforcement Learning for Foundation Models

Classical reinforcement learning approaches such as policy gradient [18], [19], [20], value learning [21], [22], [23], and preference-based RL [24], [25], [26], [27] have gained great traction in fine-tuning foundation models, especially LLMs [28], [29], [30]. Recently, there has been success in using RL to finetune foundation policies that incorporate vision, language, and actions [2], [31]. Unlike prior approaches that rely on sparse rewards or costly reward model training from labeled data, our method leverages foundation models in a zero-shot manner to provide subgoal decomposition and generalizable extrinsic reward signals for robotic fine-tuning.

C. Reward Design for Robotic Tasks

Traditional reinforcement learning (RL) relies heavily on carefully specified reward functions to guide agent behavior. In robotics, this is particularly challenging because of the sparse task success signals and high-dimensional continuous control. Poorly designed rewards often lead to misaligned incentives, suboptimal behaviors, or even unsafe strategies.

Recently, the rise of foundation models (FMs) has opened new directions for addressing the Reward Design Problem x[7]. Several works have explored how pretrained models can be leveraged to generate or shape reward functions [6], [5], [4], [7], [32]. For example, Eureka [6] uses LLMs to define reward functions by writing code, whereas VLM-RM [5] demonstrates that VLM-based similarity score can serve as a reward signal for repetitive agent patterns such as jump and walk. Leveraging the capabilities and generalizability of foundation models, this paradigm reduces reliance on hand-crafted shaping reward and mitigates the limitations of purely sparse reward signals.

Despite these advances, FM-based reward design still remains an open challenge for long-horizon tasks [33]. Prior works [2], [31] either rely on sparse reward signals or a trained reward model, both of which have limited scalability and generalizability since they either cause sample inefficiency or require task-specific manual-engineering. To address these limitations, we have developed a generalizable reward model that uses foundation models inspired by their usage in short-horizon or repetitive skills [6], [4], [5].

III. Method

A. Method Overview

Our goal is to fine-tune a pretrained robotic foundation policy using reinforcement learning algorithms such as PPO for long-horizon mobile manipulation and navigation tasks. The pretrained policy maps observations to a stochastic action distribution over the robot's action space. Since PPO requires both a policy and a value function, and the pretrained policy obtained from imitation learning does not provide a suitable critic, we introduce a value function following FLaRe [2] to enable PPO-based fine-tuning.

The overall pipeline of VLLR consists of three key components and a two-stage optimization procedure.

First, we construct a progress signal at the subgoal level, meaning that the progress increases whenever a subgoal is achieved and eventually reaches its maximum when the overall task is completed. We call it *extrinsic reward* since it'll be provided by external foundation models like LLMs and VLMs. Given a task instruction and a structured scene graph of the whole environment, a large language model (LLM) decomposes the task into an ordered sequence of grounded subgoals. A vision-language model (VLM) then evaluates the agent's visual observations with respect to the ordered subgoals, determining which subgoal the agent is

currently attempting and estimating the resulting progress toward the overall task. This provides coarse-grained supervision that reflects task-level advancement.

Second, we introduce an intrinsic reward signal derived from the pretrained policy itself. We measure policy self-certainty, which quantifies the concentration of the action distribution. Following recent work suggesting that large pretrained policies implicitly encode a form of world model, i.e., internal representations capturing regularities of environment dynamics and task structure, we interpret action distribution concentration as a proxy for how well the current state aligns with this internal model of task progression. The intrinsic reward is dense and provided in a per-step manner, which is complementary to the extrinsic reward which is coarse-grained.

Third, to ensure computational efficiency, we adopt a two-stage training procedure. In Stage I, the extrinsic reward, i.e., VLM-derived progress signal, is used to initialize the value function. In Stage II, the policy is fine-tuned using PPO with the intrinsic reward, i.e., self-certainty, and sparse task success rewards, without further VLM queries.

Together, these components form a hierarchical reward design: subgoal supervision provides coarse-level structure, intrinsic self-certainty provides fine-grained action-level guidance and sparse task success reward anchors the optimization towards the true task goal. An overview of the full pipeline is shown in Fig. 1.

B. Extrinsic Reward Design

In order to construct the reward signal that encourages progress made in completing subgoals and towards the overall goal, we adopt a four-step design. We first use LLMs to do task decomposition to provide subgoals. Secondly, the subgoals are sent to VLMs to determine which subgoal the agent is doing and estimate the overall progress. Third, we correct the raw VLM progress estimation due to its noisy nature. Fourth, we use the corrected progress estimation only at the first stage of training, i.e. value function initialization.

1) Task Decomposition with Large Language Models: Long-horizon robotic tasks like mobile manipulation are typically specified by a natural language instruction $\mathcal{I}$, which is often ambiguous and lacks step-by-step guidance for execution. In reinforcement learning, this ambiguity translates into a sparse-reward setting, where only terminal success signals are available, making credit assignment difficult.

To mitigate this issue, we introduce a structured task decomposition module based on a large language model (LLM). Formally, the LLM operates as a mapping

$$\text{LLM} : (\mathcal{G}, \mathcal{I}) \rightarrow \{g_1, g_2, \ldots, g_K\},$$

where $\mathcal{G}$ is the structured scene graph describing global environment information (e.g., rooms and objects for navigation tasks, and object relations for manipulation tasks), $\mathcal{I}$ is the high-level task instruction, and $\{g_k\}_{k=1}^{K}$ is an ordered sequence of subgoals constituting a task decomposition.

The key advantage is that the LLM has access to global structural information through $\mathcal{G}$, allowing it to reason over the entire environment rather than relying on partial observations. As a result, it can generate a coherent plan that progressively reduces spatial and semantic uncertainty.

As shown in Fig. 1, each subgoal / subtask localizes the objective and provides an intermediate verification point, effectively transforming a long-horizon sparse-reward problem into a sequence of structured short-horizon objectives.

We implement this module using a fixed prompt template that incorporates both the scene graph and the task instruction. We'll release the full prompt upon acceptance.

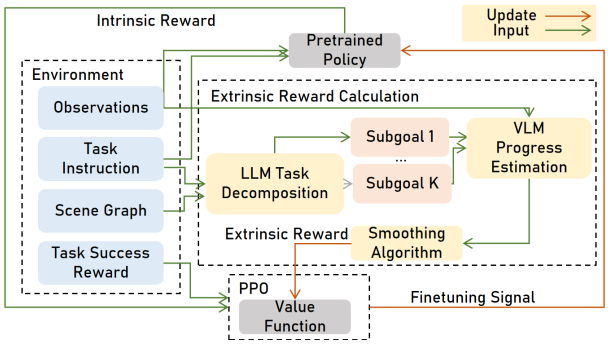

Fig. 1. Method overview. VLLR is a reward model with three components, i.e. sparse task success reward, intrinsic reward and extrinsic reward (Sec. III-D). The environment provides task instruction and a scene graph context to the LLM for task decomposition (Sec. III-B.1). Then the decomposed subgoals are fed into a VLM together with the current and last observations (Sec. III-B.2). The VLM provides a noisy progress estimate which is then smoothed out (Sec. III-B.3) and used as a reward signal for initializing the value function (Sec. III-B.4) used for PPO. The intrinsic reward is calculated using the policy's output action distribution (Sec. III-C). It is fed together with the task success signal into the PPO algorithm and then produces updates for finetuning the pretrained policy (Sec. III-D).

2) Progress Estimation with Vision-Language Models: Given the task decomposition $\{g_k\}_{k=1}^{K}$ generated by the LLM, we next transform the plan of subgoals into a learning signal that encourages subgoal completion. The key intuition is: if the agent's observation indicates that a subgoal has been achieved or that the agent is closer to the next subgoal, then the agent has made progress toward the overall task.

To operationalize this idea, we employ a vision-language model (VLM) to verify subgoal completion from visual observations. Given the ordered subgoals and the current observation, the VLM explicitly determines which subgoal the agent is currently attempting and how much progress has been made toward completing it. The VLM's output is then transformed into a scalar score that reflects the agent's progress toward the overall task, which increases as the agent successfully completes successive subgoals.

We formulate the usage of the VLM as a mapping

$$\text{VLM} : (\{g_k\}_{k=1}^{K}, o_{t-1}, o_t) \rightarrow p_t,$$

where $\{g_k\}_{k=1}^{K}$ is the ordered subgoal sequence, o_{t-1}, o_t represents the two consecutive observations at time $t-1$ and t, and $p_t \in [0, 1]$ is a scalar progress estimate indicating how much the agent has advanced toward the overall task.

Through experimentation, we found that Nova Pro, a VLM, tends to provide a progress signal that follows

a similar progression trend as the ground-truth task completion rate, although it may fluctuate across timesteps due to perception noise. We made this observation by comparing several VLMs. Specifically, we observe that CLIP tends to show no correlation with the task progress, Qwen tends to predict task completion early, whereas Nova Pro tends to predict task completion at a later time and its prediction is gradually increasing when looking at an offline video recorded using A* as provided in the CHORES benchmark [1]. We provide a representative example in Fig. 2. We adopt Nova Pro as the VLM for our method.

To derive the reward for VLLR from the progress estimation from Nova Pro, we introduce a *running maximum progress* variable $\hat{p}_t$, which tracks the highest progress estimate observed so far:

$$\hat{p}_{t+1} = \max(\hat{p}_t, p_{t+1}).$$

We then define the reward as the incremental improvement in this running maximum:

$$R_{VLM}(s_t, a_t, s_{t+1}) = \hat{p}_{t+1} - \hat{p}_t.$$

This formulation ensures that rewards are only assigned when the agent achieves new progress. However, it also makes the reward sensitive to occasional spurious high progress estimates (e.g., hallucinated peaks): once an erroneously large p_t is observed, the running maximum $\hat{p}_t$ saturates and suppresses future rewards. We address this *premature saturation* issue in the third step by stabilizing the progress estimates before computing the running maximum.

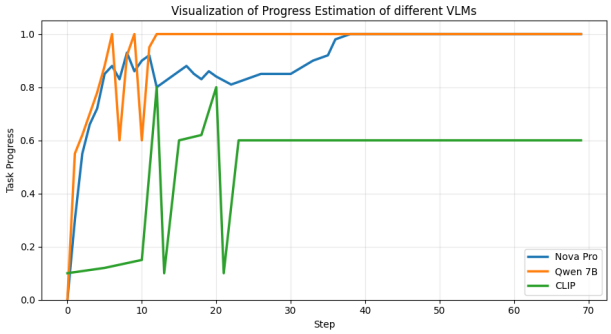

Fig. 2. We showcase an example to compare different VLM progress estimation including Qwen, Nova Pro and CLIP. The VLMs are tasked to estimate progress for finding a laptop that is not in plain sight. We found that Qwen tends to over-saturate the estimation, mostly caused by falsely recognizing the laptop, CLIP is unstable, wrong and unable to identify overall task completion, and Nova Pro is the best at progress estimation, which means it provides correct signal when seeing the laptop and approaching it. Here the rollout is collected using A* and the ground truth progress should be linearly increasing.

3) Preventing Premature Saturation of Running Maximum Progress: Once an erroneously large progress value is produced, normally due to false object recognition caused by hallucinations, it increases the *running maximum progress* $\hat{p}_t$, suppressing future rewards and effectively removing learning signals for subsequent steps.

To address this issue, we stabilize the progress estimates from Nova Pro at the trajectory level before the PPO update. Specifically, after collecting a rollout, we retrieve the

Algorithm 1 Premature saturation prevention for correcting VLM progress estimation

Initialize size of window S, pruning threshold T, buffer B that contains all progress estimations $\{p_{1:n}\}$, a new buffer that contains all processed rewards $\{r_{1:n}\}$, a running maximum progress $RMP = 0$
for $i = 1$ to n **do**
  $W = \{p_{i-S:i+S}\} \setminus \{p_i\}$
  $M = \text{Median}(W)$
  if $p_i - M > T$ **then**
    $r_i = 0$
  else
    $r_i = p_i$
  end if
end for
for $i = 1$ to n **do**
  if $r_i > RMP$ **then**
    $temp = r_i$
    $r_i = r_i - RMP$
    $RMP = temp$
  else
    $r_i = 0$
  end if
end for
return $\{r_{1:n}\}$

sequence of progress predictions

$$\{p_1, p_2, \ldots, p_T\},$$

from the PPO buffer and apply temporal consistency filtering using a sliding window of size w (e.g., $w = 5$). Genuine task progress typically evolves smoothly over time, whereas hallucinated predictions appear as isolated spikes.

For example, consider the sequence

$$\{0.1,\ 0.1,\ 0.9,\ 0.2,\ 0.2\}.$$

The value 0.9 is inconsistent with both its preceding and succeeding neighbors and is therefore identified as a spurious peak and corrected using median filtering.

Since this correction is applied after rollout collection and before the PPO update, we can leverage both past and future steps within the stored trajectory to reliably identify and suppress such artifacts. The complete procedure is detailed in Algo. 1. We also showcase an example in Fig. 3 where the raw progress estimation from Nova Pro is noisy, whereas the estimation after our correction is able to provide clear reward signals when visually verifiable cues are provided, e.g. in the figure the clock is correctly identified and Nova Pro recognizes the fact that the agent is moving closer towards the clock. We can then use this smoothed progress estimation for computing the reward as discussed in Sec. III-B.2.

4) VLM Reward as Proxy for Value Function Initialization: We observe that directly querying a VLM, e.g. LLM-based VLM like Nova Pro, at every environment step is computationally prohibitive, especially in long-horizon settings due to high latency and inference cost.

We instead use the VLM only during a short initialization phase to supervise the value function. After imitation pretraining, the VLM-derived progress signal is used to initialize the value function for 200K steps ($< 0.5\%$ of 50M total steps). No further VLM queries are performed afterward.

Unlike prior work that optimizes policies against VLM rewards throughout training [5], [4], we only use the VLM reward at Stage I for value function initialization, which then provides dense value estimates at negligible cost.

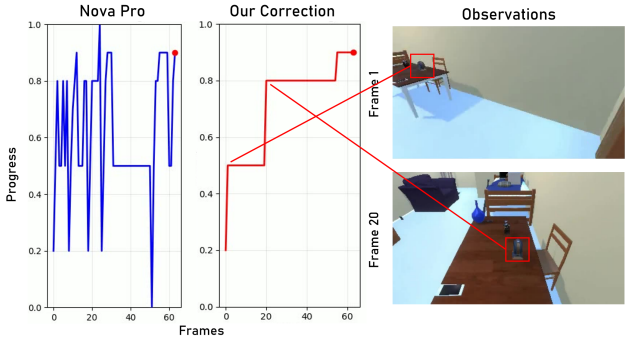

Fig. 3. We showcase an example where raw progress estimation from Nova Pro is noisy and our method is able to identify the actual progress made by the agent. The task is to find a clock and grasp it. In the first column, we can see that throughout the rollout, the progress estimation is noisy and problematic, unable to provide meaningful reward signals. In the second column, our method is able to provide three clear jumps in terms of progress estimation. The first jump rewards the fact that the robot sees the clock, the second jump rewards the robot positioning itself in front of the clock and the third jump rewards the robot actually picking up the clock. In the third column, we showcase the actual observation Nova Pro is provided. We highlight the clock in red box in frame 1 and 20. Frame 1 showcase the fact that the robot sees the clock and frame 20 demonstrates the robot positioning itself in front of the clock. We connected these two observations to the first two jumps in our progress estimation.

C. Intrinsic Reward Design

Beyond leveraging extrinsic subgoal-level supervision, we further exploit intrinsic signals from the policy model itself for per-step supervision. Recent advances in reinforcement learning for large language models (LLMs) have shown that intrinsic confidence-related metrics can serve as effective optimization signals for improving reasoning performance [8], [34]. In robotic policy learning, prior work [35] primarily relied on log-probability-based intrinsic objectives, where higher likelihood assigned to executed actions was directly used as a reward signal.

Following these insights, we adopt *self-certainty* [8] as an intrinsic reward signal for robotic policy finetuning. Rather than directly maximizing the log-probability of selected actions, self-certainty measures the overall concentration of the policy distribution. Specifically, it reflects how peaked the action distribution is: when the policy assigns high probability to a small set of actions, the distribution has lower entropy and higher self-certainty, indicating stronger confidence in the selected behavior.

Formally, self-certainty is defined as:

$$R_{SC} = -\frac{1}{|A|}\sum_{i=1}^{|A|}\log(|A|\cdot\pi_\theta(i)), \tag{1}$$

where $|A|$ denotes the size of the discrete action space and $\pi_\theta(i)$ is the probability assigned by the policy to action i.

While prior work established a positive correlation between self-certainty and reasoning correctness in LLMs [8], we extend this perspective to robotic foundation models from a structural viewpoint. This makes intuitive sense since recent studies [9] suggest that large pretrained policies implicitly encode a form of internal world model, capturing regularities of environment dynamics and task structure within their representations.

Under this view, self-certainty is not merely a measure of the concentration of the output distribution, but reflects the internal belief of how the world would evolve based on the policy's latent world representation. When the observation aligns well with the policy's internal model of task progression, the action distribution tends to be more concentrated.

D. Final Reward Model

Combining the above components, we obtain the final reward model under our two-stage optimization framework:

$$\begin{aligned} R_{\text{full}} = \alpha\,\mathbb{1}_{\{\text{Stage I}\}}\,R_{VLM} + \beta\,\mathbb{1}_{\{\text{Stage II}\}}\,R_{SC} \\ + \phi\,\mathbb{1}_{\{\text{Stage II}\}}\,R_{task}, \end{aligned} \tag{2}$$

where α, β, ϕ are weighting coefficients for different components. Here, $\mathbb{1}_{\{\text{Stage I}\}}$ and $\mathbb{1}_{\{\text{Stage II}\}}$ are indicator functions that activate the corresponding reward terms only during the specified training stage. Specifically, $\mathbb{1}_{\{\text{Stage I}\}} = 1$ during the *value initialization stage* and 0 otherwise, while $\mathbb{1}_{\{\text{Stage II}\}} = 1$ during the *policy finetuning stage* and 0 otherwise.

In Stage I, the VLM-derived reward R_{VLM} is used solely to initialize the value function, which serves as a value prior that incorporates knowledge of task hierarchy. In Stage II, the optimization relies only on the intrinsic self-certainty reward R_{SC} and the sparse task completion signal R_{task}.

IV. Experiments

We conduct our experiments on the CHORES benchmark [1], which consists of long-horizon robotic tasks that mainly focus on mobile manipulation. We observe that our method effectively guides policy optimization, resulting in both higher success rates and improved efficiency compared to baselines. Our method also enables generalization to unseen tasks in the pretraining dataset.

A. Implementation Details

We follow the same settings in FLaRe [2] for the PPO hyperparameters. For the weighting factors in Eq. 2, we set $\alpha = 1$, $\beta = 0.1$ and $\phi = 10$. Value function learning is done for 200K steps. For Object Navigation and RoomVisit we run 20M steps for training, and for the rest of the two in-distribution tasks, i.e. Fetch and Pick-up, we run 50M steps since it requires fine-grained manipulation. Object Relative Attributes Navigation and Object Affordance Navigation are trained for 50M steps. The threshold for noise detection is set as the reciprocal of the number of subtasks for the specific task and the window size is set as 9. For task decomposition we use Claude-3.7-Sonnet. For

TABLE I

SUCCESS AND EPISODE-LENGTH WEIGHTED SUCCESS (SEL) ON THE CHORES [1] BENCHMARK. OUR METHOD OUTPERFORMS THE PREVIOUS SOTA METHODS BY PROVIDING GENERALIZABLE REWARD SIGNAL ACROSS TASKS. BASELINES ARE CHOSEN FROM FLARE [2].

Success (SEL)	**IL+RL: Dense Reward**	**IL+RL: Sparse Reward**			**IL Only**
	VLLR (Ours)	FLaRe	PIRLNav	JSRL	SPOC
ObjectNav	**87.6 (77.7)**	85.0 (67.6)	20.0 (7.0)	21.0 (15.6)	55.0 (42.2)
Fetch	**70.7 (63.2)**	65.23 (54.7)	0.0 (0.0)	2.9 (2.8)	14.0 (10.5)
PickUp	**97.0 (96.4)**	91.8 (90.4)	0.0 (0.0)	50.9 (47.7)	90.1 (86.9)
RoomVisit	**68.3** (64.8)	60.87 (**65.7**)	12.5 (11.0)	19.0 (18.6)	40.5 (35.7)

TABLE II

OUR METHOD CAN SERVE AS THE REWARD SIGNAL FOR OUT-OF-DISTRIBUTION TASKS THAT ARE NEVER SEEN BY THE BASE MODEL, AND MAKE THE POLICY ACHIEVE STATE-OF-THE-ART PERFORMANCE. BASELINES ARE CHOSEN FROM FLARE [2]

Success (SEL)	VLLR (Ours)	FLaRe	Poliformer (Sp)	SPOC++	Poliformer (De)
ObjNavRel	**67.4 (61.5)**	66.3 (60.6)	6.7 (6.7)	54.5 (44.6)	36.1 (32.4)
ObjNavAff	**90.1 (78.2)**	79.7 (70.6)	35.5 (29.4)	62.4 (50.6)	53.8 (43.1)

TABLE III

ABLATION STUDY ON THREE REPRESENTATIVE TASKS. WE EXCLUDE PICKUP (NEAR-CEILING AT 97%) AND ROOMVISIT (SEL CHARACTERISTICS DISCUSSED IN SEC. IV-C) AS THEY OFFER LIMITED SIGNAL FOR COMPONENT-LEVEL COMPARISON. BASE: SPARSE REWARD ONLY; SCR: SELF-CERTAINTY REWARD; VLM: VLM PROGRESS REWARD; FULL: COMPLETE VLLR.

Success (SEL)	Base	SCR	VLM	Full
Fetch	65.23 (54.7)	69.18 (60.4)	68.00 (63.8)	70.7 (63.2)
ObjectNav	85.0 (67.6)	86.7 (76.6)	87.1 (77.5)	87.6 (77.7)
ObjNavAff	79.7 (70.6)	89.7 (76.5)	89.0 (77.1)	90.1 (78.2)

progress estimation we use Amazon Nova Pro, a lighter-weight but also best-performing VLM selected to minimize per-step inference latency: since the VLM is queried at every environment step during Stage I rollout collection, inference speed directly impacts training throughput.

B. Benchmark Configuration

The CHORES benchmark defines an observation space consisting of two egocentric RGB cameras (resolution 384×224) facing orthogonal directions, i.e., one aligned with the navigation direction and the other with the manipulator arm. At the beginning of each episode, a natural language instruction is sampled and appended to the observation, specifying the intended task. The policy has an action space of 20 discrete actions, including base translation, base rotation, arm translation, gripper rotation, as well as pickup, dropoff, task completion signal, and episode termination. Task specifications covering a diverse set of household activities. Episodes terminate either when a task is completed or when the maximum step limit is reached, in which case the attempt is marked as failed. We evaluate performance using two metrics: task **Success Rate** ($S \in \{0, 1\}$ per episode) and **Success weighted by Episode Length (SEL)** [36], defined as

$$\text{SEL} = S \cdot \frac{t_{\min}}{\max(t_{\min},\, t)}, \tag{3}$$

where $t_{\min}$ is the minimum number of steps required to complete the task, which is calculated using A* and provided by the dataset, and t is the agent's actual episode length. The benchmark is built on 191,568 houses from ProcThor [37], split into training and testing environments at a 10:1 ratio to ensure evaluation on previously unseen houses. Please refer to the original paper for details due to space limit [1], [2].

C. In-distribution tasks

We first conduct experiments on four tasks that are in the pretraining dataset of our foundation policy, SPOC [1], namely Fetch, Pick-up, RoomVisit and Object Navigation. The full task settings can be found in the Sec. V-A in the appendix. We compare our method with the SOTA method FLaRe [2], the baseline method SPOC [1], and PIRLNav [38] and JSRL [39], which apply RL finetuning with sparse task-success rewards. As shown in Tab. I, our method achieves up to 5% absolute success rate improvement over the state-of-the-art (SOTA) method. Importantly, this gain is obtained without relying on task-specific reward engineering or privileged environment annotations. We also notice significant improvements in terms of the policy's efficiency in completing the task. Compared to the baseline, the policy trained with our reward model is able to provide up to 10% absolute efficiency improvement, which demonstrates that our method is able to provide informative information about the task progress and the agent's behavior. We observe that on RoomVisit, FLaRe achieves a slightly higher SEL. We hypothesize that RoomVisit primarily emphasizes visiting multiple spatial targets, where global route ordering plays a larger role than fine-grained action efficiency. In such scenarios, small variations in path ordering can slightly affect SEL, even when overall behavioral efficiency remains comparable.

D. Out-of-distribution tasks

We further study whether the reward model we designed is generalizable enough to guide the adaptation to unseen task settings of the pretrained model. Specifically, we consider two tasks that are not in the pretraining data, namely Object Affordance Navigation and Object Relative Attribute Navigation. The full details can also be found in Sec. V-A in the appendix. For OOD tasks we additionally compare against Poliformer [1] and SPOC++ [1], which leverage privileged information in the form of human-defined task-specific dense rewards and were evaluated on ObjNavRel and ObjNavAff in the SPOC benchmark; PIRLNav and JSRL are excluded here because FLaRe has superior performance. As shown in

Tab. II, the experimental results demonstrate that our method is able to provide superior information regardless of the task setting and is able to guide the policy model to better adapt to the task goal and provides up to 10% absolute performance gain compared to SOTA on the OOD tasks.

E. Analysis

To better understand the roles of different reward components, we perform detailed ablation studies.

a) Effect of Self-Certainty on Optimization Dynamics.: We observe that incorporating self-certainty significantly accelerates convergence: the best-performing checkpoint appears much earlier than with sparse reward alone (e.g., 30M vs. 50M steps in our setting).

This suggests that self-certainty provides a dense intrinsic signal that improves credit assignment during finetuning. Given that the pretrained policy already encodes substantial task structure from imitation learning, self-certainty reinforces internally consistent predictions and speeds up alignment toward successful behaviors.

However, assigning an overly large weight to R_{SC} destabilizes training. Excess confidence can override existing value estimates, leading to catastrophic forgetting and performance collapse. Hence, careful balancing between intrinsic certainty and task-level supervision is essential.

b) Effect of VLM-Based Value Initialization.: While self-certainty primarily improves convergence speed and final success rate, VLM-based value initialization contributes more significantly to task completion efficiency. Policies initialized with VLM supervision tend to require fewer environment steps to complete successful episodes.

We attribute this to the semantic grounding provided during Stage I. The VLM-derived progress signal encodes high-level task structure, encouraging trajectories that follow the plans. As a result, the learned value function better reflects global task progression, reducing unnecessary exploration and redundant movements.

c) Success vs. Failure Behavior Patterns.: Qualitatively, policies trained with sparse reward alone tend to exhibit indecisive exploration and delayed commitment to task-relevant regions. Self-certainty-enhanced policies qualitatively appear to produce more decisive action sequences, and when combined with VLM initialization, trajectories qualitatively align more closely with task decomposition structure. These observations suggest complementary roles for the two components: self-certainty appears to enhance internal consistency and optimization speed, while VLM initialization appears to improve global task efficiency through semantic grounding. Quantitative trajectory analysis is left to future work.

d) Limitations.: Our evaluation is conducted on the CHORES benchmark [1], which is also the sole benchmark used in FLaRe [2], the primary state-of-the-art comparison and a recent ICRA 2025 accepted paper. CHORES spans a diverse range of long-horizon household tasks—from pure navigation (ObjNav, RoomVisit) to dexterous manipulation (PickUp) and combined navigation-and-manipulation (Fetch), together with two out-of-distribution tasks requiring affordance and relational reasoning (ObjNavAff, ObjNavRel). Evaluation is conducted on held-out test houses drawn from a 10:1 split of 191,568 procedurally generated ProcThor houses [2], yielding approximately 17,000 unseen test environments. The individual components of VLLR: LLM-based task decomposition, VLM progress estimation, and policy self-certainty, rely only on language-conditioned task specifications and egocentric visual observations, and are not specific to this benchmark or action space. Extending VLLR to continuous-action manipulation benchmarks is an important direction for future work.

V. CONCLUSIONS

In this paper, we present VLLR, a generalizable reward model for finetuning robotic foundation models with RL on long-horizon tasks. VLLR combines two components: (1) an extrinsic reward that measures semantic task progress using the planning and visual verification capabilities of LLMs/VLMs, which is queried only during value learning to reduce compute, and (2) an intrinsic reward derived from the policy's self-certainty, which correlates with task success. Together, they provide dense, informative signals aligned with sparse success rewards, improving sample efficiency. Experiments on the CHORES dataset demonstrate VLLR's strong generalization across both in- and out-of-distribution tasks, suggesting a promising direction for unlocking the potential of robotic foundation models.

APPENDIX

A. Task definition in CHORES

We refer the reader to the CHORES [1] benchmark for comprehensive understanding of the tasks setting.

a) Fetch: In the Fetch task, the agent is instructed to both locate and acquire a specified object. For example, *"locate a mug and pick up that mug"*.

b) Pick-Up: The Pick-Up task evaluates an agent's ability to manipulate an object that is already within its line of sight. An example instruction is *"pick up a mug"*.

c) ObjNav: Object Navigation requires the agent to go to an instance of a specified object category in the environment, such as *"find a mug"*.

d) RoomVisit: RoomVisit focuses on comprehensive environment coverage. For example, *"visit every room in this 5-room house"*.

e) ObjNavRel: Object Navigation with Relative Attributes extends ObjNav by requiring reasoning over object properties. For instance, *"find the largest apple"*

f) ObjNavAff: Object Navigation with Affordances requires the agent to interpret instructions that specify functional properties rather than object categories. For example, *"find something I can sit on"*.

ACKNOWLEDGMENT

This work was developed during the internships of SY and CQ at Amazon Robotics. SY, YX, KS were funded in part by the Army Research Laboratory (ARL) award W911QX-24-F-0049, DARPA award FA8750-23-2-1015, ONR award

N00014-23-1-2840, and ONR MURI grant N00014-25-1-2116.

REFERENCES

[1] K. Ehsani, T. Gupta, R. Hendrix, J. Salvador, L. Weihs, K.-H. Zeng, K. P. Singh, Y. Kim, W. Han, A. Herrasti *et al.*, "Spoc: Imitating shortest paths in simulation enables effective navigation and manipulation in the real world," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 16 238–16 250.

[2] J. Hu, R. Hendrix, A. Farhadi, A. Kembhavi, R. Martín-Martín, P. Stone, K.-H. Zeng, and K. Ehsani, "Flare: Achieving masterful and adaptive robot policies with large-scale reinforcement learning fine-tuning," *arXiv preprint arXiv:2409.16578*, 2024.

[3] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[4] J. Cui, T. Liu, Z. Meng, J. Yu, R. Song, W. Zhang, Y. Zhu, and S. Huang, "Grove: A generalized reward for learning open-vocabulary physical skill," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 15 781–15 790.

[5] J. Rocamonde, V. Montesinos, E. Nava, E. Perez, and D. Lindner, "Vision-language models are zero-shot reward models for reinforcement learning," *arXiv preprint arXiv:2310.12921*, 2023.

[6] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar, "Eureka: Human-level reward design via coding large language models," *arXiv preprint arXiv:2310.12931*, 2023.

[7] S. Singh, R. L. Lewis, and A. G. Barto, "Where do rewards come from," in *Proceedings of the annual conference of the cognitive science society*. Cognitive Science Society, 2009, pp. 2601–2606.

[8] Z. Kang, X. Zhao, and D. Song, "Scalable best-of-n selection for large language models via self-certainty," *arXiv preprint arXiv:2502.18581*, 2025.

[9] J. Richens, D. Abel, A. Bellot, and T. Everitt, "General agents contain world models," *arXiv preprint arXiv:2506.01622*, 2025.

[10] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[11] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," *Advances in neural information processing systems*, vol. 36, pp. 34 892–34 916, 2023.

[12] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[13] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang *et al.*, "Gr00t n1: An open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025.

[14] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[15] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning*. PMLR, 2023, pp. 2165–2183.

[16] J. Huang, S. Yong, X. Ma, X. Linghu, P. Li, Y. Wang, Q. Li, S.-C. Zhu, B. Jia, and S. Huang, "An embodied generalist agent in 3d world," *arXiv preprint arXiv:2311.12871*, 2023.

[17] Y. Fan, P. Ding, S. Bai, X. Tong, Y. Zhu, H. Lu, F. Dai, W. Zhao, Y. Liu, S. Huang *et al.*, "Long-vla: Unleashing long-horizon capability of vision language action model for robot manipulation," *arXiv preprint arXiv:2508.19958*, 2025.

[18] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[19] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[22] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[23] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. Pmlr, 2018, pp. 1861–1870.

[24] C. Wirth, R. Akrour, G. Neumann, and J. Fürnkranz, "A survey of preference-based reinforcement learning methods," *Journal of Machine Learning Research*, vol. 18, no. 136, pp. 1–46, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-634.html

[25] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement: The tamer framework," in *Proceedings of the fifth international conference on Knowledge capture*, 2009, pp. 9–16.

[26] ——, "Reinforcement learning from simultaneous human and mdp reward." in *AAMAS*, vol. 1004. Valencia, 2012, pp. 475–482.

[27] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

[28] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan *et al.*, "Training a helpful and harmless assistant with reinforcement learning from human feedback," *arXiv preprint arXiv:2204.05862*, 2022.

[29] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," *Advances in neural information processing systems*, vol. 36, pp. 53 728–53 741, 2023.

[30] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.

[31] G. Lu, W. Guo, C. Zhang, Y. Zhou, H. Jiang, Z. Gao, Y. Tang, and Z. Wang, "Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning," *arXiv preprint arXiv:2505.18719*, 2025.

[32] Y. J. Ma, J. Hejna, C. Fu, D. Shah, J. Liang, Z. Xu, S. Kirmani, P. Xu, D. Driess, T. Xiao *et al.*, "Vision language models are in-context value learners," in *The Thirteenth International Conference on Learning Representations*, 2024.

[33] M. Lin, S. Shi, Y. Guo, B. Chalaki, V. Tadiparthi, E. M. Pari, S. Stepputtis, J. P. Campbell, and K. P. Sycara, "Navigating noisy feedback: Enhancing reinforcement learning with error-prone language models," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024, pp. 16 002–16 014.

[34] X. Zhao, Z. Kang, A. Feng, S. Levine, and D. Song, "Learning to reason without external rewards," *arXiv preprint arXiv:2505.19590*, 2025.

[35] Z. Zhang, K. Zheng, Z. Chen, J. Jang, Y. Li, S. Han, C. Wang, M. Ding, D. Fox, and H. Yao, "Grape: Generalizing robot policy via preference alignment," *arXiv preprint arXiv:2411.19309*, 2024.

[36] A. Eftekhar, K.-H. Zeng, J. Duan, A. Farhadi, A. Kembhavi, and R. Krishna, "Selective visual representations improve convergence and generalization for embodied ai," *arXiv preprint arXiv:2311.04193*, 2023.

[37] M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, K. Ehsani, J. Salvador, W. Han, E. Kolve, A. Kembhavi, and R. Mottaghi, "Procthor: Large-scale embodied ai using procedural generation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5982–5994, 2022.

[38] R. Ramrakhya, D. Batra, E. Wijmans, and A. Das, "Pirlnav: Pretraining with imitation and rl finetuning for objectnav," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 896–17 906.

[39] I. Uchendu, T. Xiao, Y. Lu, B. Zhu, M. Yan, J. Simon, M. Bennice, C. Fu, C. Ma, J. Jiao *et al.*, "Jump-start reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 34 556–34 583.