Language Reward Modulation for Pretraining Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Using learned reward functions (LRFs) as a means to solve sparse-reward reinforcement learning (RL) tasks has yielded some steady progress in task-complexity through the years. In this work, we question whether today's LRFs are best-suited as a direct replacement for task rewards. Instead, we propose leveraging the capabilities of LRFs as a pretraining signal for RL. Concretely, we propose **LA**nguage Reward **M**odulated **P**retraining (LAMP) which leverages the zero-shot capabilities of Vision-Language Models (VLMs) as a *pretraining* utility for RL as opposed to a downstream task reward. LAMP uses a frozen, pretrained VLM to scalably generate noisy, albeit shaped exploration rewards by computing the contrastive alignment between a highly diverse collection of language instructions and the image observations of an agent in its pretraining environment. LAMP optimizes these rewards in conjunction with standard novelty-seeking exploration rewards with reinforcement learning to acquire a language-conditioned, pretrained policy. Our VLM pretraining approach, which is a departure from previous attempts to use LRFs, can warmstart sample-efficient learning on robot manipulation tasks in RLBench.

1 Introduction

A longstanding challenge in reinforcement learning is specifying reward functions. Extensive domain knowledge and ad-hoc tuning are often required in order to manually design rewards that "just work." However, such rewards can be highly uninterpretable and riddled with cryptic mathematical expressions and constants. Furthermore, hand-crafted reward functions are often over-engineered to the domain in which they were designed, failing to generalize to new agents and new environments (Shah et al., 2022; Langosco et al., 2022). As a result, a long history of foundational work in Inverse Reinforcement Learning (IRL) (Abbeel & Ng, 2004; Ng & Russell, 2000; Kim et al., 2003; Peng et al., 2021; Escontrela et al., 2022) has produced an abundance of methods for learning rewards from demonstration data assumed to optimal under the desired reward function. However, learned reward functions are also notorious for noise and reward misspecification errors (Amodei et al., 2016; Amodei & Clark, 2016) which can render them highly unreliable for learning robust policies with reinforcement learning. This is especially problematic in more complex task domains such as robotic manipulation, particularly when in-domain data for learning the reward function is limited.

While acknowledging that learned reward functions are subject to potential errors, we hypothesize that they may be effectively employed to facilitate exploration during the lower-stakes pretraining stage of training. During pretraining, we generally desire a scalable means of generating diverse behaviors to warmstart a broad range of possible downstream tasks. LRFs are an appealing means for supervising these behaviors since they do not rely on human-design and carry the potential to scale with dataset diversity. Despite this potential, obtaining LRFs that generalize to new domains is non-trivial (Shah et al., 2022). Notably, however, large-pretrained models have shown impressive zero-shot generalization capabilities that enable them to be readily applied in unseen domains. Indeed, large pretrained VLMs have shown recent successes in reward specification for task learning by computing alignment scores between the image observations of an agent and a language input

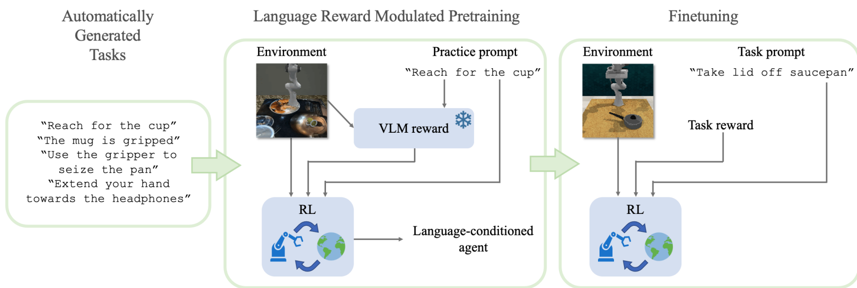

Figure 1: **LAMP Framework**. Given a diverse set of tasks generated by hand or by a LLM, we extract VLM rewards for language-conditioned RL pretraining. At finetuning time, we condition the agent on the new task language embedding and finetune on the task reward.

describing the desired task (Cui et al., 2022; Mahmoudieh et al., 2022). While these methods adopt similar reward mispecification shortcomings to other LRFs, they come with the novel and relatively under-explored property of being a scalable means of generating many *different* rewards by simply prompting with different language instructions. This property is particularly compatible with the assumptions of RL pretraining where we desire to learn general-purpose behaviors with high coverage of environment affordances, require minimal human supervision, and do not require pretrained behaviors to transfer zero-shot to fully solve downstream tasks. Instead of relying on noisy VLM LRFs to train task-specific experts, can we instead use them as a tool for pretraining a general-purpose agent?

In this work, we investigate how to use the flexibility of VLMs as a means of scalable reward generation to *pretrain* an RL agent for accelerated downstream learning. We propose **LA**nguage Reward **M**odulated **P**retraning (LAMP), a method for pretraining diverse policies by optimizing VLM parameterized rewards. Our core insight is that instead of scripting rewards or relying solely on general unsupervised objectives to produce them, we can instead query a VLM with highly diverse language prompts and the visual observations of the agent to generate diverse, shaped pretraining rewards. We augment these rewards with intrinsic rewards from Plan2Explore (Sekar et al., 2020), a novelty-seeking unsupervised RL algorithm, resulting in an objective that biases exploration towards semantically meaningful visual affordances. A simple language-conditioned, multitask reinforcement learning algorithm optimizes these rewards resulting in a language-conditioned policy that can be finetuned for accelerated downstream task learning. We demonstrate that by pretraining with VLM rewards in a visually complex environment with diverse objects, we can learn a general-purpose policy that more effectively reduces the sample-complexity of downstream RL. We train LAMP in a pretraining environment with realistic visual textures and challenging randomization and evaluate downstream performance on unseen RLBench tasks. We also analyze the influence of various prompting techniques and frozen VLMs on the performance of the pretrained policy.

2 Related Work

Pretraining for RL Following on the successes of pretraining in vision and language, a number of approaches have grown in interest for pretraining generalist RL agents in order to reduce sample-complexity on unseen tasks. Classical works in option learning (Sutton et al., 1999; Stolle & Precup, 2002) and more recent approaches in skill discovery such as (Sharma et al., 2019; Gregor et al., 2016; Park et al., 2023; Eysenbach et al., 2018) look to pretrain skill policies that can be finetuned to downstream tasks. Exploration RL algorithms such as (Burda et al., 2018; Sekar et al., 2020; Pathak et al., 2017; Liu & Abbeel, 2021) use unsupervised objectives to encourage policies to learn exploratory behaviors. Works such as (Xiao et al., 2022; Nair et al., 2022; Radosavovic et al., 2023) leverage pretrained vision encoders to accelerate RL from pixels. LAMP combines a large-pretrained VLM with exploration-based RL to guide exploration towards meaningful behaviors.

Inverse RL from human video Inverse reinforcement learning (IRL) (Arora & Doshi, 2021; Ziebart et al., 2008; Abbeel & Ng, 2004; Ng & Russell, 2000) proposes a number of approaches to address the challenges associated with learning reward functions from demonstrations. A number of more recent works focus on inverse RL from video datasets of human interaction, (Chen et al., 2021; Zakka et al., 2021; Sermanet et al., 2016; 2017) which are often more readily available than in-domain demonstrations. These methods rely on perceptual metrics such as goal-image similarity and trajectory similarity to formulate rewards but require task-specific paired data. Other methods such as (Ma et al., 2023b;a) make weaker assumptions on the task-specificity of the human video dataset and thus can leverage "in-the-wild" data and exhibit stronger domain generalization. LAMP similarly exploits "in-the-wild" video data via a frozen, pretrained VLM but focuses on leveraging language to flexibly modulate the VLM and generate diverse rewards.

3 Background

Reinforcement learning with vision-language reward We consider the reinforcement learning (RL) framework where an agent receives an observation o_t from an environment and chooses an action a_t with a policy π to interact with the environment. Then the agent receives an extrinsic reward $r_t^{\mathtt{e}}$ and a next observation o_{t+1} from the environment. The goal of RL is to train the policy to maximize the expected return defined as a cumulative sum of the reward with a discount factor γ, *i.e.,* $\mathcal{R}_t = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r^{\mathtt{e}}(o_{t+k}, a_{t+k})]$. In sparse reward tasks, the extrinsic reward $r^{\mathtt{e}}$ becomes non-zero only when the task successfully terminates, making it difficult to learn policies that complete the target tasks. To address this, recent approaches have proposed to use the vision-language alignment score from either a bespoke and in-domain or large-scale vision-language model (VLM) as a reward (Fan et al., 2022; Cui et al., 2022; Du et al., 2023b; Mahmoudieh et al., 2022; Sontakke et al., 2023; Du et al., 2023a; Fan et al., 2022; Shao et al., 2020). Formally, let $\mathbf{x} := \{x_1, ..., x_M\}$ be a text that describes the task consisting of M tokens, F_ϕ be a visual feature encoder, and L_α be a language encoder. Given a sequence of transitions $\{o_i, a_i, r_i^{\mathtt{e}}, o_{i+1}\}_{i=1}^N$, the key idea is to use the distance between visual representations $F_\phi(o_i)$ and text representations $L_\alpha(\mathbf{x})$ as an intrinsic reward, which is defined as $r_i^{\mathtt{int}} = D\left(F_\phi(o_i), L_\alpha(\mathbf{x})\right)$, where D can be an arbitrary distance metric such as cosine similarity or L2 distance. This intuitively can be seen as representing the extent to which the current observation is close to achieving the task specified by the text.

Video-language models The vision encoders of video-language models have been successfully employed as semantic feature extractors that enable downstream learning on a variety of domains including standard prediction and classification tasks as well as, more recently, decision making and control (Xu et al., 2021; Wang et al., 2022). Notably, R3M, has lead to improvements in the data-efficiency of imitation learning in real-world robotics domains (Nair et al., 2022). R3M extracts semantic representations from the large-scale Ego4D dataset of language annotated egocentric human videos (Grauman et al., 2022). The language input is processed by L_α, a pretrained DistilBERT transformer architecture (Sanh et al., 2019) that aggregates the embeddings of each word in the instruction and the images are encoded with a pretrained ResNet-18 F_ϕ. A video-language alignment loss encourages the image representations $F_\phi(\cdot)$ to capture visual semantics by extracting image features that aid in predicting the associated language annotations, which are embedded by $L_\alpha(\mathbf{x})$. In particular, R3M trains $G_\theta(F_\phi(o_1), F_\phi(o_i), L_\alpha(\mathbf{x}))$ to score whether the language $\mathbf{x}$ explains the behavior from image o_1 to image o_i. The score function is trained simultaneously to the representations described above with a contrastive loss that encourages scores to increase over the course of the video and scores to be higher for correct pairings of video and language than incorrect ones.

4 Method

We present **LA**nguage Reward **M**odulated **P**retraining (LAMP), a simple yet effective framework for pretraining reinforcement learning with intrinsic rewards modulated with language instructions. LAMP consists of two stages: (i) a task-agnostic RL pretraining phase that trains policies to maximize the VLM-based rewards and (ii) a downstream task finetuning phase that adapts pre-trained policies to solve the target tasks by maximizing the task reward. In this section, we first describe how we define our intrinsic reward (see Section 4.1), how we pretrain policies (see Section 4.2), and how we adapt the policies to downstream tasks (see Section 4.3). We provide an overview of LAMP in Figure 2 and pseudocode in Algorithm 1 of the appendix.

4.1 Language Reward Modulation

VLM score as a reward To extract pretraining reward signals for RL, we elect to use the R3M score as a source of rewards for most of the experiments presented. Specifically, we expect R3M to be well-suited for providing shaped rewards because its representations are trained on videos. However, we also find that CLIP (Radford et al., 2021), which is trained on static image data, can also work well with the appropriate reward parameterization as shown in Section 5. We define our VLM reward using the R3M score as below:

$$r_i^{\mathtt{VLM}} = G_\theta(F_\phi(o_1), F_\phi(o_i), L_\alpha(\mathbf{x}))$$

where G_θ denotes the score predictor in R3M. Intuitively, this reward measures how o_i is making a progress from o_1 towards achieving the tasks specified by natural language instruction $\mathbf{x}$.

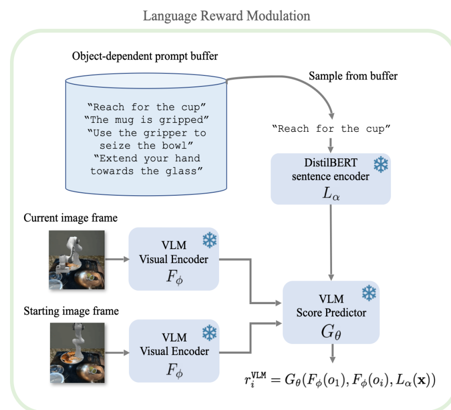

Figure 2: **LAMP Method**. We use R3M for our VLM-based rewards. We query the VLM score predictor for pixel and language alignment, which is pre-trained on the Ego4D dataset. The reward model is frozen.

Rewards with diverse language prompts To fully exploit the language understanding of VLMs, we propose to query them with a diversity of texts describing a diversity of objectives, as opposed to computing the reward by repeatedly using a single instruction. Specifically, we obtain diverse, semantic rewards modulated by language, generating diverse sets of language instructions for each task and use them for prompting the model. Given an instruction template, we query ChatGPT[1] for a diverse set of language instructions with a focus on two categories: imperative instructions and statements of completion (*e.g.* move the mug vs. the mug is moved). Given that large-scale video datasets are predominantly human-centric, we obtain prompts that are human-centric, robot centric, as well as ambiguous (*e.g.* the robot arm moved the mug vs. use your hand to move the mug vs. reach toward the mug and move it). Moreover, we augment the instructions by querying for synonym nouns. By inputting diverse language instructions from the dataset along with the agent's image observations, we effectively modulate the frozen, pretrained R3M reward and produce diverse semantic rewards that are grounded in the visual environment of the agent.

4.2 Language-Conditioned Pretraining

While several recent works have shown that rewards from VLMs can be used for training RL agents (Fan et al., 2022; Cui et al., 2022), it is still questionable whether these rewards can serve as a sufficiently reliable signal for inducing the intended behavior. As we validate in Figure 3, for three different VLM-paramterized reward models, the optimization signal is highly noisy and in some instances does not correspond well to the task semantics. As a result, using these rewards directly as a downstream reward with a single language instruction for a given task is unlikely to reliably result in robust policy learning with RL as found in (Ma et al., 2023b;a). In this work, we instead propose to leverage such rewards from VLMs as a pretraining signal for RL policies, utilizing the knowledge captured within large VLMs for scripting diverse and useful exploration rewards.

[1] https://chat.openai.com

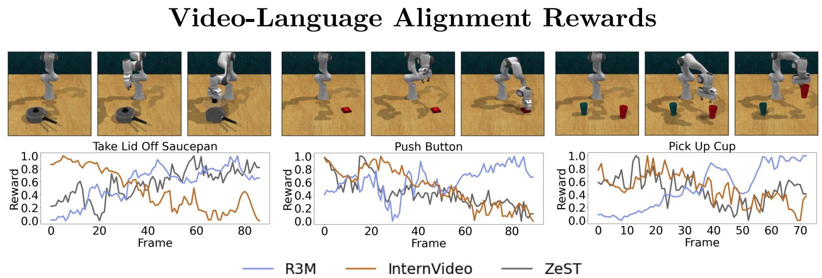

Figure 3: Video-Language alignment scores from R3M, InternVideo (Wang et al., 2022), and ZeST on RLBench downstream tasks plotted over an expert episode. Rewards are highly noisy and do not increase smoothly throughout the episode. Optimizing this signal with RL is unlikely to lead to stable solutions, and thus we instead use rewards as an exploration signal during pretraining.

Pretraining environment To learn diverse behaviors that can be transferred to various downstream tasks, we design a set of tasks with realistic visuals and diverse objects. Specifically, we build a custom environment based on the RLBench simulation toolkit (James et al., 2020). In order to simulate a realistic visual scene, we download images from the Ego4D dataset (Grauman et al., 2022) and overlay them as textures on the tabletop and background of the environment (see Figure 2). To produce diverse objects and affordances, we import ShapeNet (Chang et al., 2015) object meshes into the environment. Both the visual textures and the objects are randomized every episode of training. The pretraining environments do include the original RLBench textures, however, they do not include the finetuning RLBench tasks as shown in Figure 8.

Objective Because the VLM reward can be seen as measuring the extent to which the agent is closer to solving the task (see Section 4.1), it can be readily be combined with novelty-seeking unsupervised RL methods that optimize both extrinsic and intrinsic rewards. Therefore, to incentivize exploration, we combine the VLM reward with the novelty score from a separate exploration technique. Specifically, we consider Plan2Explore (Sekar et al., 2020) that utilizes the disagreement between future latent state predictions as a novelty score. Let this novelty-based score be $r_i^{\texttt{P2E}}$. We then train our pretraining agent to maximize the following weighted sum of rewards:

$$r_i^{\texttt{LAMP}} = \alpha \cdot r_i^{\texttt{P2E}} + (1-\alpha) \cdot r_i^{\texttt{VLM}} \tag{1}$$

where α is a hyperparameter that balances the two rewards. By combining this novelty-based reward with the VLM reward, we encourage the agent to efficiently explore its environment but with an additional bias towards interacting with the semantically meaningful affordances. We found that an α value of 0.9 works quite well across all the tasks evaluated.

Pretraining pipeline During task-agnostic pretraining, the agent is deployed in a language-conditioned MDP where there are no environment task rewards $r_i^{\texttt{e}}$. For the underlying RL algorithm, we use Masked World Models (MWM) (Seo et al., 2022), an off-policy, model-based method with architectural inductive biases suitable for fine-grained robot manipulation, because we found that more standard alternatives such as PPO (Schulman et al., 2017), SAC (Haarnoja et al., 2018), and Dreamer (Hafner et al., 2023) were unable to reliably solve all of the evaluation RLBench tasks from ground truth scripted rewards. Every episode, our method randomly samples some language prompt $\mathbf{x}$ from the generated dataset as specified in Section 4.1. Then we condition the MDP and agent on the respective embedding $L_\alpha(\mathbf{x})$ such that each rolled out transition in the episode can be expressed as $(o_i, a_i, o_{i+1}, L_\alpha(\mathbf{x}))$. After each episode is collected, we compute the rewards for each transition by embedding the observations with the R3M visual encoder F_ϕ and then applying the R3M score predictor G_θ - afterwards adding the data to the replay buffer. We then sample batches of transitions from the buffer, augment the reward with the Plan2Explore intrinsic reward, and make reinforcement learning updates to a language-conditioned policy, critic, and world model. By the

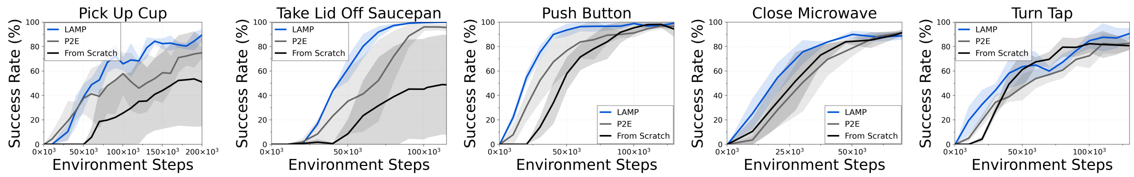

Figure 4: Finetuning performance on visual robotic manipulation tasks in RLBench. We include results with additional unsupervised RL baselines in the supplementary material.

conclusion of pretraining, we obtain a language-conditioned policy capable of bootstrapping diverse behaviors specified by the language $\mathbf{x}$.

4.3 Downstream Task Adaptation

In order to evaluate the quality of the pretrained skills, we evaluate on downstream reinforcement learning tasks from the original RLBench task suite, including the ones pictured in Figure 3, with scripted task rewards r_i^e. Since we have learned a language-conditioned policy, we simply select a language instruction $\mathbf{x}_{\mathrm{ft}}$ roughly corresponding to the downstream task semantics in order to condition the pretrained agent. We remark that an additional advantage of LAMP is its use of language as a task-specifier, which enables this simplicity of zero-shot selection of a policy to finetune (Adeniji et al., 2022). We fix this language instruction selection for the entirety of task learning and finetune all RL agent model components except the critic, which we linear probe for training stability.

5 Experiments

Environment details As previously mentioned in Section 4.2, we consider domain-randomized environments for pre-training (see Figure 8 for examples). Specifically, our pretraining environments consist of 96 domain-randomized environments with different Ego4D textures overlayed on the table, walls, and floor. We also sample the environments with default RLBench environment textures with probability of 0.2. We do not include any of the downstream evaluation tasks that the agent finetunes on in the pretraining setup. The pretraining tasks are exclusively comprised of ShapeNet objects.

For finetuning, we implement a shaped reward function based on the ground truth simulator state and train the agent to optimize this signal instead of the pretraining reward. We use the exact scenes and tasks released in RLBench in order to encourage reproducibility, notably keeping the default background and table textures fixed throughout the course of training. The downstream RLBench tasks include those pictured in Figure 3 and are unseen during pretraining. We use a 4-dimensional continuous action space where the first three dimensions denote end-effector positional displacements and the last controls the gripper action. We fix the end-effector rotation and thus select tasks that can be solved without changing the orientation of the end-effector.

Baselines As a baseline, we first consider a randomly initialized MWM agent trained from scratch to evaluate the benefit of pretraining. In order to evaluate the benefit of pretraining with LAMP that modulates the reward with language, we also consider Plan2Explore (Sekar et al., 2020) as a baseline, which is a novelty-seeking method that explores based on model-uncertainty. Our method, LAMP, is the combination of the VLM reward and the Plan2Explore reward with a fixed α value of 0.9 across tasks as described in Equation (1)

Downstream Finetuning Results Across tasks in Figure 4, we find that training a randomly initialized agent from scratch on a new task exhibits high-sample complexity in order to learn a performant policy. The solid line and shaded region represent mean and standard deviation across 3 seeds for all experiments. Across most RLBench tasks, Plan2Explore, which employs purely unsupervised exploration, exceeds the performance of training from scratch. LAMP outperforms or is competitive with Plan2Explore and consistently outperforms training from scratch. We hypothesize that this is because by pretraining with a VLM reward, LAMP biases the agent to explore efficiently in ways that are semantic meaningful and avoids spending time investigating spurious novelty. It is also possible that by optimizing more diverse rewards during pretraining, the agent learns representations that allow it to quickly adapt to the unseen task reward during finetuning. We further note that the VLM rewards in isolation can lead to effective pretraining as we demonstrate in Figure 7. Solely optimizing the VLM rewards (LAMP w/o Plan2Explore) is able to achieve strong performance, however, we find that the combination of VLM and Plan2Explore rewards yields the best overall performance.

What is the best Language Prompting style? A convenience of using VLMs trained on internet-scale data is the diversity of language we can query for plausibly infinite reward functions. We evaluate how important it is that the prompts be aligned with possible tasks in the pretraining environment. We also investigate whether providing a variety of prompt semantics mitigates the pathology where imperfections in the VLM reward for a single prompt are exploited. The 6 language prompting styles ablated are: **Prompt Style 1**: `Pick up the [NOUN].`, **Prompt Style 2**: `[RELEVANT VERB] and [SYNONYM NOUN].`, **Prompt Style 3**: `[RELEVANT VERB] and [RANDOM NOUN].`, **Prompt Style 4**: `[IRRELEVANT VERB] and [SYNONYM NOUN].`, **Prompt Style 5**: `[IRRELEVANT VERB] and [RANDOM NOUN].`, **Prompt Style 6**: `[Snippets from Shakespeare].` For Prompt Styles 1-5, we compare the effect of using relevant and irrelevant nouns and verbs, though all remain action-based and task-relevant. For Prompt Style 6, we select snippets of Shakespeare text to see the effect of pretraining on rewards generated from completely out of distribution and implausible tasks.

In Figure 5 (Left), we compare the finetuning results of Prompt Styles 1-5, which are action-based prompts on the task "Pick Up Cup". We find that for this task, Prompt Style 2, which pretrains on semantically similar but diverse wordings of prompts, is most successful. In addition, Prompt Style 1, which pretrains on very simple instructions that are similar to the pretraining task, finetunes efficiently, as well. For our main experiments, we choose Prompt 2 based on both its strong finetuning performance as well as increased diversity compared to Style 1.

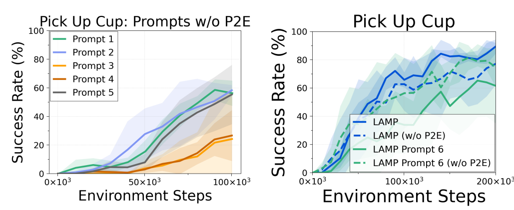

Figure 5: Finetuning performance on RL-Bench task. (Left) Effect of pretraining with rewards from different language prompting styles. Language prompts focus on action-based tasks. (Right) Effect of pretraining on action-based prompts (Lang 2) and random prompts (Lang 6).

In Figure 5 (Right), we also compare the performance of our best action-based prompt, Prompt 2, with a non-action-based prompt, Prompt 6. In this figure, we also investigate the effect of the auxiliary exploration objectives. While LAMP Prompt 6 (w/o Plan2Explore) and LAMP Prompt 2 (w/o Plan2Explore) perform similarly, we notice that adding in the exploration objectives dramatically decreases the finetuning effectiveness of LAMP Prompt 6. We hypothesize that both exploration coverage and exploration efficiency during pretraining play an important role. By separately increasing exploration coverage through Plan2Explore, the quality of the VLM rewards becomes more important for focusing the auxiliary exploration objective on useful exploratory behaviors. Thus, LAMP Prompt 2, which incorporates Plan2Explore, is trained on higher-quality, more relevant rewards, and can explore more efficiently during pretraining, and therefore enables more effective finetuning. Overall, the differences in pretraining with different action-based prompting styles is not extreme, suggesting that LAMP is robust to different prompting strategies, and providing diverse instructions can be an effective way of pretraining an agent.

How much does the Reward Weighting or VLM matter? We provide an ablation of the α reward weighting term for the joint pretraining reward introduced in Equation 1. We isolate the relative contributions of the Plan2Explore and LAMP rewards by setting the alpha value to 0 and 1. We find that, while either option performs somewhat similarly, the synergy of the two rewards

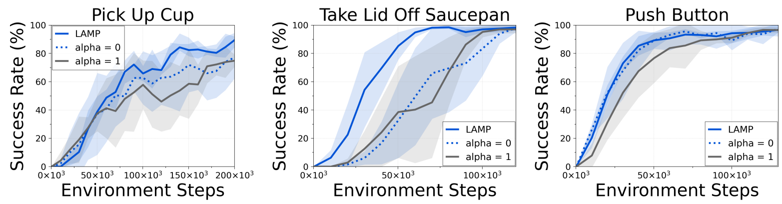

Figure 6: We find that LAMP is robust to the reward weighting hyperparameter alpha. Notably, with purely VLM rewards (alpha = 0), LAMP obtains competitive performance across all tasks evaluated. However, the synergy of the two rewards achieves the strongest performs.

achieves the highest performance, particularly on the "Take Lid Off Saucepan Task." We conduct this ablation on additional tasks as well as perform a sweep of the reward weighting hyperparameter in Appendix 13.

We evaluate LAMP using another popular off-the-shelf VLM. CLIP Radford et al. (2021), which is trained on static image-caption pairs, can serve as a reward model by extracting the cosine similarity between text feature and image feature changes as presented as ZeST in (Cui et al., 2022). Following ZeST, in order to express feature displacements, we assign context features in image and language space as s_0, the initial image in an episode, and $\mathbf{x_0}$, a language prompt that inverts the action (open microwave inverts close microwave) described in the desired instruction $\mathbf{x}$. In particular, we use the following reward parameterization,

$$r_i = (F_\phi(s_i) - F_\phi(s_0)) \cdot (L_\alpha(\mathbf{x}) - L_\alpha(\mathbf{x_0})).$$

where F_ϕ is the CLIP pretrained image encoder, L_α is the CLIP pretrained language encoder, and the task reward r_i is defined as the dot product of the visual and language delta embeddings.

We show finetuning results on the RLBench downstream tasks in Figure 7. While R3M seems to lead to consistently strong performance, we observe that LAMP pretraining with CLIP rewards performs similarly well, confirming the robustness of our pipeline with respect to the choice of the specific VLM. Additional tasks are in Figure 14 in the Appendix.

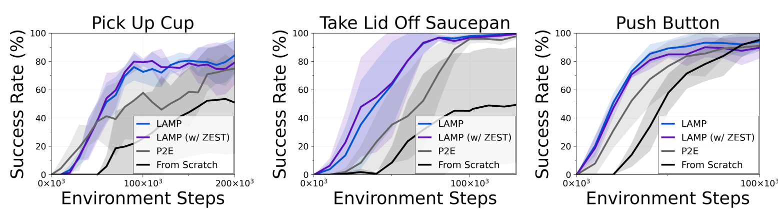

Figure 7: Finetuning performance on visual robotic manipulation tasks in RLBench. We observe that CLIP rewards can also work well with LAMP components.

6 Discussion

In this work, we present LAMP, an algorithm for leveraging frozen VLMs to pretrain reinforcement learning agents. We observe a number of limitations of using LRFs for task reward specifcation in RL, and propose a new setting and methodology that is more accommodating of these limitations. We demonstrate that by leveraging the flexibility and zero-shot generalization capabilities of VLMs, we can easily produce diverse rewards during pretraining that encourage the learning of exploratory behaviors useful for many downstream tasks. Furthermore, we show that VLM parameterized rewards exhibit synergies with novelty-seeking RL pretraning methods. Overall, we see LAMP as an indication of the promise of leveraging large pretrained VLMs for pretraining behaviors in challenging environments.

References

Pieter Abbeel and Andrew Ng. Apprenticeship learning via inverse reinforcement learning. *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*, 09 2004. doi: 10.1007/978-0-387-30164-8_417.

Ademi Adeniji, Amber Xie, and Pieter Abbeel. Skill-based reinforcement learning with intrinsic reward matching, 2022.

Dario Amodei and Jack Clark. Faulty reward functions in the wild. https://blog.openai.com/faulty-reward-functions/, 2016. Accessed: May 15, 2023.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016. URL http://arxiv.org/abs/1606.06565.

Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. *CoRR*, abs/1810.12894, 2018. URL http://arxiv.org/abs/1810.12894.

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository, 2015.

Annie S. Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from "in-the-wild" human videos, 2021.

Yuchen Cui, Scott Niekum, Abhinav Gupta, Vikash Kumar, and Aravind Rajeswaran. Can foundation models perform zero-shot task specification for robot manipulation?, 2022.

Yuqing Du, Ksenia Konyushkova, Misha Denil, Akhil Raju, Jessica Landon, Felix Hill, Nando de Freitas, and Serkan Cabi. Vision-language models as success detectors, 2023a.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models, 2023b.

Alejandro Escontrela, Xue Bin Peng, Wenhao Yu, Tingnan Zhang, Atil Iscen, Ken Goldberg, and Pieter Abbeel. Adversarial motion priors make good substitutes for complex reward functions. *International Conference on Intelligent Robots and Systems*, 2022. doi: 10.48550/ARXIV.2203.15103.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge, 2022.

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh Kumar Ramakrishnan, Fiona Ryan, Jayant Sharma, Michael Wray, Mengmeng Xu, Eric Zhongcong Xu, Chen Zhao, Siddhant Bansal, Dhruv Batra, Vincent Cartillier, Sean Crane, Tien Do, Morrie Doulaty, Akshay Erapalli, Christoph Feichtenhofer, Adriano Fragomeni, Qichen Fu, Abrham Gebreselasie, Cristina Gonzalez, James Hillis, Xuhua Huang, Yifei Huang, Wenqi Jia, Weslie Khoo, Jachym Kolar, Satwik Kottur, Anurag Kumar, Federico Landini, Chao Li, Yanghao Li, Zhenqiang Li, Karttikeya Mangalam, Raghava Modhugu, Jonathan Munro, Tullie Murrell, Takumi Nishiyasu, Will Price, Paola Ruiz Puentes, Merey Ramazanova, Leda Sari, Kiran Somasundaram, Audrey Southerland, Yusuke Sugano, Ruijie Tao, Minh Vo, Yuchen Wang, Xindi Wu, Takuma Yagi, Ziwei Zhao, Yunyi Zhu, Pablo Arbelaez, David Crandall, Dima Damen, Giovanni Maria Farinella, Christian Fuegen, Bernard Ghanem, Vamsi Krishna Ithapu, C. V. Jawahar, Hanbyul Joo, Kris Kitani, Haizhou Li, Richard Newcombe, Aude Oliva, Hyun Soo Park, James M. Rehg, Yoichi Sato, Jianbo Shi, Mike Zheng Shou, Antonio Torralba, Lorenzo Torresani, Mingfei Yan, and Jitendra Malik. Ego4d: Around the world in 3,000 hours of egocentric video, 2022.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018. URL http://arxiv.org/abs/1801.01290.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2023.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020.

H. Kim, Michael Jordan, Shankar Sastry, and Andrew Ng. Autonomous helicopter flight via reinforcement learning. In S. Thrun, L. Saul, and B. Schölkopf (eds.), *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2003. URL https://proceedings.neurips.cc/paper_files/paper/2003/file/b427426b8acd2c2e53827970f2c2f526-Paper.pdf.

Lauro Langosco Di Langosco, Jack Koch, Lee D Sharkey, Jacob Pfau, and David Krueger. Goal misgeneralization in deep reinforcement learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 12004–12019. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/langosco22a.html.

Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. *Advances in Neural Information Processing Systems*, 34:18459–18473, 2021.

Yecheng Jason Ma, William Liang, Vaidehi Som, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control, 2023a.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training, 2023b.

Parsa Mahmoudieh, Deepak Pathak, and Trevor Darrell. Zero-shot reward specification via grounded natural language. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 14743–14752. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/mahmoudieh22a.html.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation, 2022.

Andrew Ng and Stuart Russell. Algorithms for inverse reinforcement learning. *ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning*, 05 2000.

Seohong Park, Kimin Lee, Youngwoon Lee, and Pieter Abbeel. Controllability-aware unsupervised skill discovery. *arXiv preprint arXiv:2302.05103*, 2023.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP: adversarial motion priors for stylized physics-based character control. *CoRR*, abs/2104.02180, 2021. URL https://arxiv.org/abs/2104.02180.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. URL https://arxiv.org/abs/2103.00020.

Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. In *Conference on Robot Learning*, pp. 416–426. PMLR, 2023.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019. URL http://arxiv.org/abs/1910.01108.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. *CoRR*, abs/2005.05960, 2020. URL https://arxiv.org/abs/2005.05960.

Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control, 2022.

Pierre Sermanet, Kelvin Xu, and Sergey Levine. Unsupervised perceptual rewards for imitation learning. *CoRR*, abs/1612.06699, 2016. URL http://arxiv.org/abs/1612.06699.

Pierre Sermanet, Corey Lynch, Jasmine Hsu, and Sergey Levine. Time-contrastive networks: Self-supervised learning from multi-view observation. *CoRR*, abs/1704.06888, 2017. URL http://arxiv.org/abs/1704.06888.

Rohin Shah, Vikrant Varma, Ramana Kumar, Mary Phuong, Victoria Krakovna, Jonathan Uesato, and Zac Kenton. Goal misgeneralization: Why correct specifications aren't enough for correct goals, 2022.

Lin Shao, Hema Koppula, and Jinjun Xiong. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*. Robotics: Science and Systems Foundation, 2020.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

Sumedh A Sontakke, Jesse Zhang, Sébastien M. R. Arnold, Karl Pertsch, Erdem Bıyık, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. Roboclip: One demonstration is enough to learn robot policies, 2023.

Martin Stolle and Doina Precup. Learning options in reinforcement learning. In Sven Koenig and Robert C. Holte (eds.), *Abstraction, Reformulation, and Approximation*, pp. 212–223, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1): 181–211, 1999. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(99)00052-1. URL https://www.sciencedirect.com/science/article/pii/S0004370299000521.

Yi Wang, Kunchang Li, Yizhuo Li, Yinan He, Bingkun Huang, Zhiyu Zhao, Hongjie Zhang, Jilan Xu, Yi Liu, Zun Wang, Sen Xing, Guo Chen, Junting Pan, Jiashuo Yu, Yali Wang, Limin Wang, and Yu Qiao. Internvideo: General video foundation models via generative and discriminative learning, 2022.

Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer. Videoclip: Contrastive pre-training for zero-shot video-text understanding, 2021.

Kevin Zakka, Andy Zeng, Pete Florence, Jonathan Tompson, Jeannette Bohg, and Debidatta Dwibedi. XIRL: cross-embodiment inverse reinforcement learning. *CoRR*, abs/2106.03911, 2021. URL https://arxiv.org/abs/2106.03911.

Brian Ziebart, Andrew Maas, J. Bagnell, and Anind Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, pp. 1433–1438, 01 2008.

A Masked World Models

For our experiments, we use Masked World Models (MWM) (Seo et al., 2022) as our underlying RL algorithm. MWM is a model-based RL framework that aims to decouple visual representation learning and dynamics learning. In contrast to prior algorithms that learn world models in an end-to-end manner, MWM separately learns a visual encoder via masked autoencoding (He et al., 2021) and a world model that reconstructs frozen autoencoder representations. We build LAMP upon the official implementation provided by authors (`https://github.com/younggyoseo/MV-MWM`) which supports experimentation on RLBench (James et al., 2020). In Table 7, 8, and 9 we provide all relevant hyperparameters.

B Video-Language Models

R3M For the R3M (Nair et al., 2022) VLM in LAMP, we use the official implementation (`https://github.com/facebookresearch/r3m`). We use the ResNet-18 visual backbone and preserve all default hyperparameters. To encode the language prompts, we use the DistilBERT 'base-uncased' model (`https://huggingface.co/distilbert-base-uncased`) from the transformers (`https://pypi.org/project/transformers/`) package as used in the R3M implementation.

InternVideo For the InternVideo (Wang et al., 2022) VLM in LAMP we use the official implementation (`https://github.com/OpenGVLab/InternVideo`). We use the "B/16" model and pretrained weights provided by the authors for embedding the images and text.

For InternVideo (Wang et al., 2022) we match the style of alignment score computation used in training and inference. We use the following reward parameterization:

$$r_i = F_\phi([s_{i//8}, s_{2*i//8}, ..., s_{8*i//8}]) \cdot L_\alpha(l).$$

where 8 frames evenly spaced from the agent's entire history are featurized by the visual encoder F_ϕ.

ZeST For the CLIP (Radford et al., 2021) VLM used in ZeST (Cui et al., 2022) for LAMP, we use the official OpenAI CLIP model (`https://github.com/openai/CLIP`). We use the "ViT-B/32" release of the CLIP visual encoder for embedding images and the CLIP text encoder for embedding langauge prompts.

C Experimental Details

C.1 Language Prompting Types

We ablate on 6 different prompt styles, with the structures defined in Section 5. To construct Prompt Style 1, we replace the `[NOUN]` in the `Pick up the [NOUN].` prompt with the ShapeNet name. To construct Prompt Style 6, we sample from random Shakespeare phrases listed below. For the remaining Prompt Styles, we sample a verb structure, either `IRRELEVANT` or `RELEVANT`; and a noun, either `RANDOM` or `SYNONYM`. Both verb structures are included in Table 2. We include examples of random nouns and synonym nouns for a sample object; nouns and synonyms for all objects are ommitted for space, but they can be found in the code.

Verb structures and synonyms were curated from ChatGPT, with filtering afterward to select most relevant verbs. Random nouns were taken from the synonyms. We provide the automatically generated language prompt datasets for each language prompting scheme used during LAMP pretraining. Prompts to ChatGPT are included in Table 1.

Algorithm 1 Language Reward Modulated Pretraining (LAMP)

Initialize Masked World Models (MWM) parameters, Load pretrained DistilBERT L_α, Load pretrained R3M visual encoder F_ϕ, Load pretrained R3M score predictor G_θ, Initialize replay buffer $\mathcal{B} \leftarrow 0$, Prefill language prompt buffer $\mathcal{B}^l$, Prefill synonym buffer $\mathcal{B}^s$
for each episode **do**
  Randomize scene textures by sampling among Ego4D and original RLBench textures
  Sample ShapeNet Objects to place in scene
  Sample language prompt $\mathbf{x}$ from $\mathcal{B}^l$ (e.g., `Pick up the [NOUN]`)
  Replace `[NOUN]` in $\mathbf{x}$ by sampling a synonym from $\mathcal{B}^s$ for a random object in the scene
  Process the prompt via DistilBERT to obtain language embeddings $L_\alpha(\mathbf{x})$
  Collect episode transitions with $\pi(a|(s, L_\alpha(\mathbf{x}))$
  Assign LAMP rewards to all time steps (in parallel) by embedding frames with F_ϕ and querying the R3M score predictor G_θ
  Add all episode transitions to $\mathcal{B}$
  Update MWM and Plan2Explore parameters as in (Seo et al., 2022; Sekar et al., 2020) by sampling transitions from $\mathcal{B}$ and augmenting LAMP rewards with novelty bonus to train agent
end for

Table 1: **Using an LLM to generate verb structures and nouns**. We query ChatGPT with prompts to create a set of diverse tasks.

Generating Verb Structures	**Generating Synonyms**
Give me a list of 40 task variations that present an interesting task for a person to do in a home or kitchen scenario. Examples should not be complicated, and should be possible to do very quickly, within a minute or so. These should be simple tasks that are interesting and diverse, but EASY. Tasks should be atomic and very general. For example: 1. Reach for the mug 2. Open the microwave 3. Wipe the table clean 4. Water the flowers	Please give me 40 synonyms for bowl Example: 1. bowl 2. soup bowl 3. dish

C.2 ShapeNet Objects

In Figure 9 we provide images of some of the ShapeNet object assets used in the pretraining environments. We provide some examples of generated pretraining environments in Figure 8.

C.3 Ego4D Textures

In Figure 10 we provide the textures extracted from Ego4D we overlayed on the pretraining environments.

C.4 LAMP Algorithm

We describe the LAMP algorithm in detail in Algorithm 1.

Table 2: **Verb Structures**. We include the different verb structures used during pretraining.

Relevant Verb	**Irrelevant Verb**
Pick up the [NOUN]	The [NOUN] is seized
Lift the [NOUN] with your hands	The [NOUN] is clutched
Hold the [NOUN] in your grasp	The [NOUN] is gripped
Take hold of the [NOUN] and raise it	The [NOUN] is firmly grasped
Grasp the [NOUN] firmly and lift it up	The [NOUN] is tightly held
Raise the [NOUN] by picking it up	The [NOUN] is firmly caught
Retrieve the [NOUN] and hold it up	The [NOUN] is securely clasped
Lift the [NOUN] by gripping it	The [NOUN] is rotated
Seize the [NOUN] and raise it off the surface	The [NOUN] has been flipped
Hold onto the [NOUN] and lift it up	The [NOUN] has been knotted
The [NOUN] is lifted up	The [NOUN] has been folded
The [NOUN] is picked up off the ground	The [NOUN] has been rinsed
The [NOUN] is raised up by hand	The [NOUN] has been filled
The [NOUN] is grasped and lifted up	The [NOUN] is shaken
The [NOUN] is taken up by hand	The [NOUN] has been scooped
The [NOUN] is retrieved and lifted up	The [NOUN] is poured
The [NOUN] is lifted off its surface	The [NOUN] has been scrubbed
The [NOUN] is elevated by being picked up	The [NOUN] is tilted
The [NOUN] is hoisted up by hand	The [NOUN] has been heated
The [NOUN] is scooped up and lifted	Reach for the [NOUN]
The [NOUN] is lifted by the hand	Grasp at the [NOUN]
The [NOUN] is grasped and picked up	Stretch out to touch the [NOUN]
The [NOUN] is raised by the palm	Move your arm towards the [NOUN]
The [NOUN] is taken up by the fingers	Use the gripper to rinse the [NOUN]
The [NOUN] is held and lifted up	Position the end effector to fold the [NOUN]
The [NOUN] is lifted off the surface by the arm	Reach out the robotic arm to wipe the [NOUN]
The [NOUN] is picked up and held by the wrist	Utilize the gripper to seize the [NOUN]
The [NOUN] is scooped up by the palm and lifted	Guide the robotic arm to obtain the [NOUN]
The [NOUN] is elevated by the hand	Maneuver the end effector to lift up the [NOUN]
The [NOUN] is taken up by the fingers of the hand	Extend your hand towards the [NOUN]
The [NOUN] is grasped and raised	Reach out your hand to acquire the [NOUN]
The [NOUN] is lifted by the gripper	Guide your arm to rotate the [NOUN]
The end effector picks up the [NOUN]	Maneuver your hand to shake up the [NOUN]
The arm lifts the [NOUN]	Flip the [NOUN]
The [NOUN] is held aloft by the robotic hand	Tap the [NOUN]
The robotic gripper secures the [NOUN]	Fold the [NOUN]
The [NOUN] is lifted off the surface by the robotic arm	Rotate the [NOUN]
The robotic manipulator seizes and elevates the [NOUN]	Brush the [NOUN]
The robotic end effector clasps and hoists the [NOUN]	Twist the [NOUN]
The [NOUN] is taken up by the robotic gripper	Wipe the [NOUN]

Table 3: **Prompt Style 6**. Snippets from Shakespeare.

Holla, Barnardo. BARNARDO Say, what, is Horatio there? HORATIO A piece of him. Welcome, Horatio.—Welcome, good Marcellus. HORATIO What, has this thing appeared again tonight? BARNARDO I have seen nothing. MARCELLUS Horatio says 'tis but our fantasy And will not let belief take hold of him Touching this dreaded sight twice seen of us. Therefore I have entreated him along With us to watch the minutes of this night, That, if again this apparition come, He may approve our eyes and speak to it. Tush, tush, 'twill not appear. Sit down awhile, How now, Horatio, you tremble and look pale. Is not this something more than fantasy? What think you on 't? At least the whisper goes so: our last king, Whose image even but now appeared to us, Was, as you know, by Fortinbras of Norway, Thereto pricked on by a most emulate pride, Dared to the combat; in which our valiant Hamlet (For so this side of our known world esteemed him) Did slay this Fortinbras, who by a sealed compact, Well ratified by law and heraldry, Did forfeit, with his life, all those his lands Which he stood seized of, to the conqueror. Against the which a moiety competent Was gagèd by our king, which had returned To the inheritance of Fortinbras Had he been vanquisher, as, by the same comart And carriage of the article designed, His fell to Hamlet. Now, sir, young Fortinbras, Of unimprovèd mettle hot and full, Hath in the skirts of Norway here and there Sharked up a list of lawless resolutes BARNARDO

Table 4: **Nouns**. Synonym and random nouns for "bag."

Synonym Noun	Random Noun
bag	cap
handbag	hat
purse	snapback
clutch	faucet
tote	tap
backpack	vase
knapsack	flask
satchel	earphone
shoulder bag	earpiece
duffel bag	knife
messenger bag	blade
grip	laptop
briefcase	notebook
pouch	vase
fanny pack	flowerpot
drawstring	telephone
beach bag	flip phone
grocery shop	handle
shopping bag	lever
gift bag	gift bag
lunch bag	lunch bag
laptop bag	laptop bag
travel bag	travel bag

Figure 8: We pretrain on domain-randomized environments with Ego4D textures, occasionally sampling the default, non-randomized RLBench texture. The RLBench *tasks* are not included in the pretraining environments.

C.5 Compute

We used a NVIDIA DGX A100 GPUs for all experimentation. Pretraining time for 300k steps takes 60-72 hours; finetuning for 100k takes 16-19 hours.

D Limitations

A limitation of LAMP is its reliance on performing inference of the VLM model many times throughout the course of pretraining in order to generate rewards. Inference speed for larger, more powerful VLMs may be slow, bottlenecking the speed of the pretraning process. Another limitation is that LAMP does not address the long-horizon sequencing setting where we might be interested in finetuning agents conditioned on many language prompts. We leave these as intriguing directions for future work.

Figure 9: Examples of ShapeNet object assets used during the pretraining phase.

E Additional Experiments and Ablations

E.1 Finetuning Results with Instruction Tuning

We include additional finetuning results for the "Phone On Base" RLBench task in Figure 11. We find that by tuning the language instruction used to condition the model, we further increase performance. We provide results involving selecting a frozen language instruction from a set of many semantically similar generated instructions for finetuning. We consider 9 of these new generated prompts in addition to the original task name based prompt. We then select the language prompt corresponding to the policy that achieves the highest zero-shot evaluation return when rolled out. This simple tuning step provides us with increases in performance on the new "Phone on Base" task. The optimal instructions tuned for each of the three seeds include:

1. The arm is picking up the phone and placing it on the base
2. The robot arm grasps the phone and sets it down
3. The robot gripper picks up the phone and places it on the base

E.2 Random Network Distillation

We experiment with Random Network Distillation (Burda et al., 2018), an additional unsupervised reinforcement learning algorithm, as an additional baseline and report results in 12. We find that for the "Pick Up Cup" and "Push Button" tasks Plan2Explore is a stronger baseline. While in the case for the "Take Lid Off Saucepan" task, RND does manage to outperform Plan2Explore, LAMP exhibits stronger performance than either baseline. We report all relevant hyperparameters in Table 10.

E.3 Alpha Reward Weighting Ablation

We provide ablation of the reward weighting on additional tasks as well as provide a sweep of the reward weighting parameter on the Take Lid Off Saucepan task in Figure 13. We find that the method is somewhat robust to the choice of α, however, larger values in general work better.

E.4 VLM Model Ablation Additional Tasks

We include the Close Microwave and Turn Tap tasks for the CLIP (ZeST) VLM version of LAMP in Figure 14.

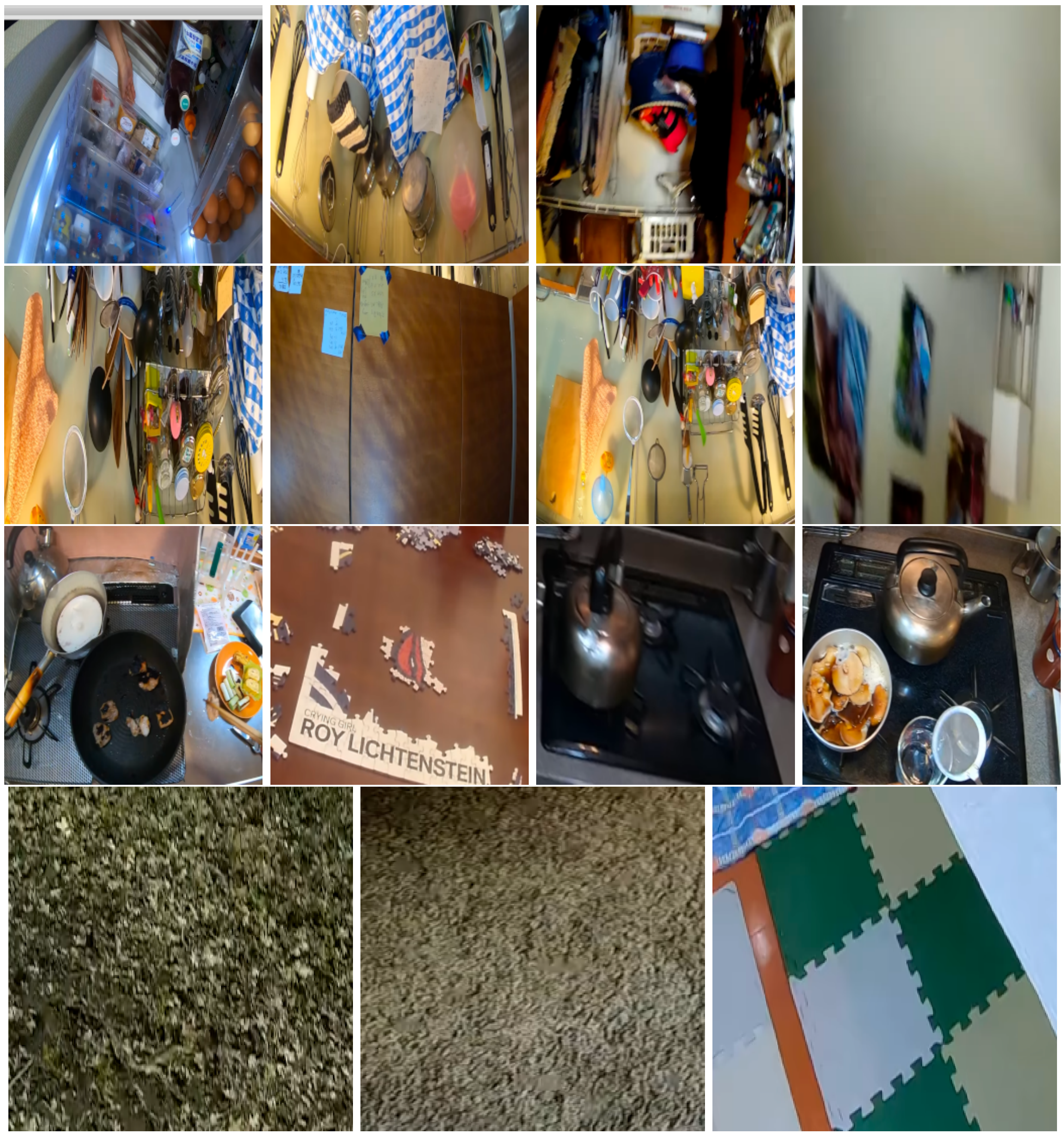

Figure 10: Textures cropped out of Ego4D videos and overlayed to the RLBench scene. The textures on the first two rows were overlayed to the walls, those on the third row to the table, and those on the fourth row to the floor.

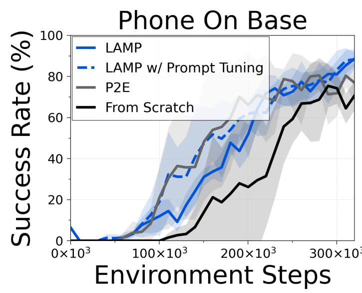

Figure 11: Finetuning Result on Phone On Base

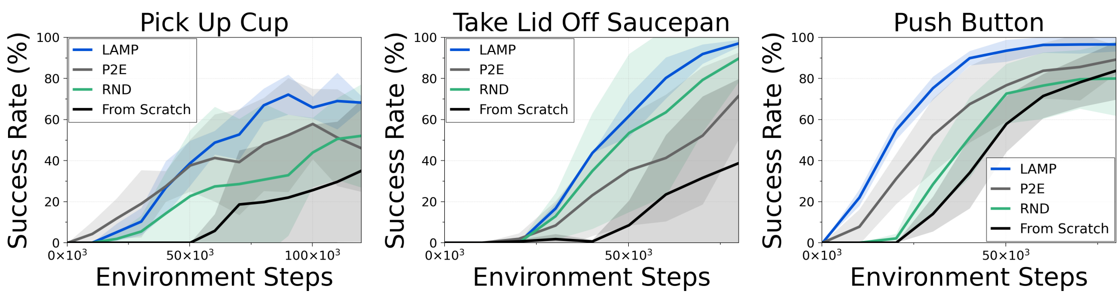

Figure 12: Finetuning performance on visual robotic manipulation tasks in RLBench. The solid line and shaded region represent mean and standard deviation across 3 seeds.

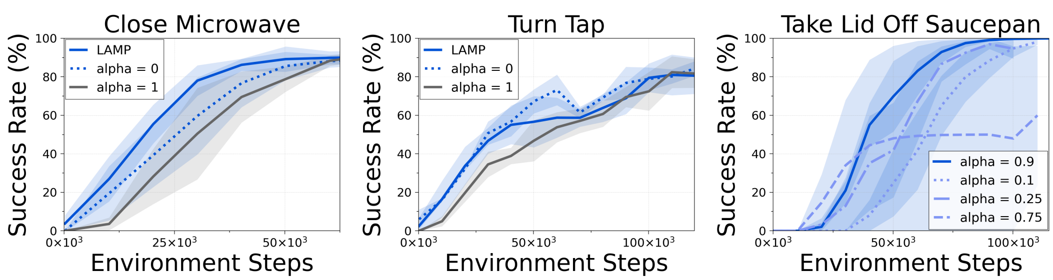

Figure 13: We find that LAMP is robust to the reward weighting hyperparameter alpha. Notably, with purely VLM rewards (alpha = 0), LAMP obtains competitive performance across all tasks evaluated. However, the synergy of the two rewards achieves the strongest performs.

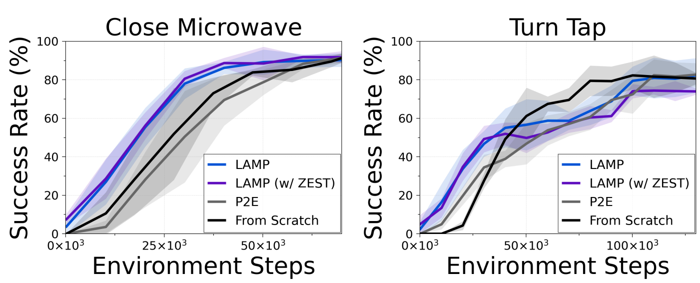

Figure 14: Finetuning performance on additional visual robotic manipulation tasks in RLBench. The solid line and shaded region represent mean and standard deviation across 3 seeds. We observe that CLIP rewards can also work well with LAMP components.

F Hyperparameters

Table 5: Plan2Explore Hyperparameters

Parameter	**Value**
Plan2Explore	False
Exploration Intrinsic Scale	0.9
Exploration Extrinsic Scale	0.1
Exploration Optimization	Optimization: Adam
	Learning Rate: 3e-4
	Epsilon: 1e-5
	Clip: 100
	Weight Decay: 1e-6
Exploration Head	Layers: [512, 512, 512, 512]
	Activation: ELU
	Normalization: None
	Distribution: MSE
Exploration Reward Normalization	Momentum: 1.0
	Scale: 1.0
	Epsilon: 1e-8
Disaggregation Target	Stochastic
Disaggregation Log	False
Disaggregation Models	10
Disaggregation Offset	1
Disaggregation Action Condition	True
Exploration Model Loss	KL

Table 6: PTMae Hyperparameters

Parameter	**Value**
MAE Image Width Size	224
MAE Image Height Size	224
WM Flat VIT Image Height Size	7
WM Flat VIT Image Width Size	7
MAE State Prediction	False
WM Flat VIT Input Channels	768
WM Flat VIT Embedding Dimension	128
MAE Average	True

Table 7: MAE Hyperparameters

Parameter	**Value**
Camera Keys	'image front|image wrist'
Mask Ratio	0.95
MAE	Image Height Size: 128
	Image Width Size: 128
	Patch Size: 16
	Embedding Dimension: 256
	Depth: 8
	Number of Heads: 4
	Decoder Embedding Dimension: 256
	Decoder Depth: 6
	Decoder Number of Heads: 4
	Reward Prediction: True
	Early Convolution: True
	State Prediction: True
	Input Channels: 3
	Number of Cameras: 0
	State Dimension: 10
	View Masking: True
	Control Input: 'front wrist'
WM Flat VIT	Image Height Size: 8
	Image Width Size: 8
	Patch Size: 1
	Embedding Dimension: 128
	Depth: 2
	Number of Heads: 4
	Decoder Embedding Dimension: 128
	Decoder Depth: 2
	Decoder Number of Heads: 4
	Input Channels: 256
	State Prediction: False
Image Time Size	4
Use ImageNet MAE	False
MAE Chunk	1
MAE Average	False

Table 8: World Model Hyperparameters

Parameter	**Value**
Grad Heads	[Reward, Discount]
Predictive Discount	True
RSSM	Action Free: False
	Hidden: 1024
	Deterministic: 1024
	Stochastic: 32
	Discrete: 32
	Activation: ELU
	Normalization: None
	Stochastic Activation: Sigmoid2
	Minimum Standard Deviation: 0.1
Reward Head	Layers: [512, 512, 512, 512]
	Activation: ELU
	Normalization: None
	Distribution: Symlog
Discount Head	Layers: [512, 512, 512, 512]
	Activation: ELU
	Normalization: None
	Distribution: Binary
Loss Scales	Feature: 1.0
	KL: 1.0
	Reward: 1.0
	Discount: 1.0
	Proprio: 1.0
	MAE Reward: 1.0
KL	Scale: 1.0
KL Minloss	0.1
KL Balance	0.8
Model Optimization	Optimization: Adam
	Learning Rate: 3e-4
	Epsilon: 1e-5
	Clip: 100.0
	Weight Decay: 1e-6
	Weight Decay Pattern: ‘kernel’
	Warmup: 0
MAE Optimization	Optimization: Adam
	Learning Rate: 3e-4
	Epsilon: 1e-5
	Clip: 100.0
	Weight Decay: 1e-6
	Warmup: 2500

Table 9: Actor Critic Hyperparameters

Parameter	**Value**
Actor	Layers: [512, 512, 512, 512] Activation: ELU Normalization: None Distribution: Trunc_Normal Minimum Standard Deviation: 0.1
Critic	Layers: [512, 512, 512, 512] Activation: ELU Normalization: None Distribution: MSE
Actor Optimization	Optimization: Adam Learning Rate: 1e-4 Epsilon: 1e-5 Clip: 100.0 Weight Decay: 1e-6 Weight Decay Pattern: 'kernel' Warmup: 0
Critic Optimization	Optimization: Adam Learning Rate: 1e-4 Epsilon: 1e-5 Clip: 100.0 Weight Decay: 1e-6 Weight Decay Pattern: 'kernel' Warmup: 0
Discount	0.99
Discount Lambda	0.95
Image Horizon	15
Actor Grad	Dynamics
Actor Grad Mix	0.1
Actor Entropy	Scale: 1e-4
Slow Target	True
Slow Target Update	100
Slow Target Fraction	1
Slow Baseline	True
Reward Normalization	Momentum: 0.99 Scale: 1.0 Epsilon: 1e-8

Table 10: Random Network Distillation Hyperparameters

Parameter	**Value**
Embedding Dimension	512
Hidden Dimension	256
Optimizer	Adam Learning Rate: 3e-4 Epsilon: 1e-5 Clip: 100.0 Weight Decay: 1e-6 Warmup: 2500