RELOOP: PROGRESS SUPERVISION FROM LLM-GENERATED REWARD PROGRAM EXECUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) can produce executable reward programs from task descriptions, but the downstream reward interface typically exposes only each program's scalar return. Distances and contact predicates computed inside the program may reveal differences in task progress even when scalar rewards coincide. We introduce RELOOP, which records these runtime values and maps them to a progress potential and trace state. Potential changes supply a training signal, while a frontier bonus and memory of observed success use past traces to provide additional signals. RELOOP leaves the generated reward logic and policy observations unchanged and requires no further LLM calls. Across eight manipulation tasks, it improves mean final success by 17.1 percentage points over direct optimization of the same programs and by 7.0 points over a control matched for reward scale and success feedback. A control built from the scalar return favors the structured interface on average. On three tasks, the potential term retains the endpoint advantage over the matched control without the two memory terms. RELOOP thus turns intermediate values from a generated reward program into progress supervision during policy training.

1 INTRODUCTION

Large language models (LLMs) can translate task descriptions and environment APIs into executable reward programs (Yu et al., 2023; Xie et al., 2024; Ma et al., 2024). A manipulation reward may compute the distance from a gripper to an object, a contact predicate, the distance from the object to its goal, and a stability measure before combining them into a scalar. During policy optimization, the reward interface usually exposes only that return. Two executions can therefore have the same reward even when their intermediate values indicate different task progress.

Recent methods use feedback from policy training to search for or refine reward programs (Ma et al., 2024; Zeng et al., 2024; Zhang et al., 2025a; Gao et al., 2025). Other work makes progress explicit through automata, supplied stages, demonstrations, or progress functions written by LLMs (Icarte et al., 2018; 2022; Mu et al., 2024; Zhang et al., 2025c; Chen et al., 2026). We study a distinct *post-generation interface*: whether intermediate computations in an already available reward program can provide training signals beyond its scalar return, without another generation step.

RELOOP instruments a generated reward program once, recording runtime values and predicates without changing its return. A deterministic grounder maps these events to four channels: reach, interaction, goal attainment, and stability. Changes in the resulting potential supply a training signal; exceeding the episode's previous maximum by a margin triggers a frontier bonus. Visit counts for discretized trace states temper this bonus; their transition and success statistics provide an additional shaping signal. The wrapper supplies the shaped reward to PPO or SAC without changing policy observations or learner update rules (Figure 1).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

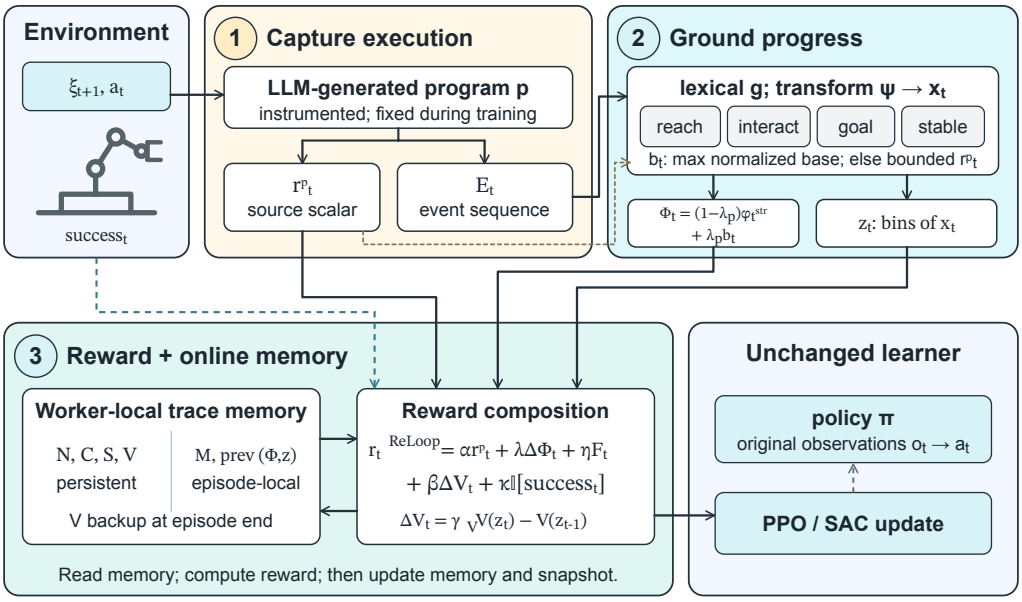


Figure 1: **ReLoop reward pathway. 1. Capture:** Each transition runs the fixed, instrumented LLM-generated program once, producing scalar r_t^p and events E_t. **2. Ground:** Events yield channels x_t, potential Φ_t, and binned state z_t; a bounded transform of r_t^p supplies b_t if no base event survives. **3. Shape and update:** The wrapper reads memory local to each worker before computing r_t^{ReLoop}, then records visits, transitions, and environment success; V is backed up at episode end. The figure uses $\Delta\Phi_t = \Phi_t - \Phi_{t-1}$ and $\Delta V_t = \gamma_V V(z_t) - V(z_{t-1})$. The unchanged observation–action loop is abbreviated in the policy panel; PPO/SAC updates are unchanged.

Across eight manipulation tasks from ManiSkill2 and Meta-World, RELOOP improves mean final success by 17.1 percentage points over direct optimization of the same reward programs. Against a control matched for reward scale and success feedback, the mean gain is 7.0 points. PickCube shows the clearest matched gain, from 59.3% to 88.7% final success, with an AUC gain of 24.5 points. An extended scalar-return control favors the structured interface on average across the eight tasks. On three diagnostic tasks, potential change retains the endpoint advantage over Matched after removing the frontier and success memory terms. These comparisons support runtime intermediates as a source of progress supervision for the evaluated programs.

Our contributions are:

- We identify runtime intermediates of LLM-generated reward programs as a post-generation source of progress supervision.
- We introduce RELOOP, which grounds execution events into an online potential and trace state for training without revising reward code or policy observations.
- We compare the interface with direct optimization, a matched control, and a scalar return control across eight tasks, then probe potential change on three tasks.

2 RELATED WORK

Language models for reward design. Language models support reward design through learned signals and executable code (Mahmoudieh et al., 2022; Yu et al., 2023; Xie et al., 2024). Text2Reward uses human feedback to refine generated Python rewards (Xie et al., 2024). Eureka tracks reward components during training and summarizes them to guide later code revision (Ma et al., 2024). Other methods refine or align rewards, select candidates online, or search candidate populations and trees (Zeng et al., 2024; Zhang et al., 2025a; Li et al., 2025b; Gao et al., 2025; Li et al., 2025a).

Language models can also guide exploration or define auxiliary rewards (Du et al., 2023; Triantafyllidis et al., 2024; Ye et al., 2025); diffusion-augmented agents offer a separate route to exploration and transfer (Palo et al., 2025). Demonstrations and language feedback support reward learning and adaptation (Huang et al., 2023; Chen et al., 2025; Zhang et al., 2025b; Yang et al., 2025). Visual and perceptual signals provide reward estimates or shaping (Zhao et al., 2025; Ayalew et al., 2025; Yu et al., 2025; Lee et al., 2026), while reward-and-policy pretraining targets autonomous real-world RL (Yang et al., 2024). PoRSE couples reward shaping with exploration; LIMEN searches observation mappings and reward functions jointly (Chang et al., 2026; Jaswal et al., 2026). In contrast to revising or replacing the supplied program, RELOOP uses its intermediates at each step to construct training signals within one policy training run.

Explicit task structure and progress models. Explicit progress representations provide task structure beyond a scalar reward value. Reward machines encode propositions and transitions, including hierarchical compositions (Icarte et al., 2018; 2022; Furelos-Blanco et al., 2023). Recent work learns or exploits such structure under uncertain semantics (Verginis et al., 2024; Li et al., 2024; Lin & Zhang, 2025), and formal task specifications can induce hierarchical potential shaping (Berducci et al., 2025; Kwon et al., 2025). ARM-FM generates reward machines, executable labeling functions, and language descriptions used to condition policies (Creus Castanyer et al., 2026). DrS learns dense rewards from stage information, while PRM4RL constructs progress functions from LLM-generated subtask plans (Mu et al., 2024; Zhang et al., 2025c). ProgressCounts discretizes progress features written by LLMs and uses visit counts to generate intrinsic rewards (Sarukkai et al., 2024). SARM and general process reward models learn progress estimates from robot demonstrations or visual observations (Chen et al., 2026; Tan et al., 2026). More broadly, work on process rewards distinguishes progress signals from terminal outcome scores (Setlur et al., 2025). RELOOP builds its representation from intermediate values already evaluated by a reward program. This changes the source of progress information while retaining the policy's original observations.

Reward shaping and exploration. Potential-based shaping preserves optimal policies under conditions on discounting, potential updates, and episode boundaries (Ng et al., 1999; Wiewiora, 2003; Devlin & Kudenko, 2012; Grześ, 2017). Bonuses based on visit counts encourage exploration (Ostrovski et al., 2017), and RUDDER redistributes delayed returns using learned return decompositions (Arjona-Medina et al., 2019). RELOOP draws on potential differences, visit counts, and tabular backups to construct rewards over its runtime representation. Its frontier bonus is restricted to new potential maxima within an episode, whereas ProgressCounts rewards visits to discretized progress states (Sarukkai et al., 2024). The complete RELOOP reward is history dependent, so classical guarantees of policy invariance do not directly apply.

Program instrumentation. Dynamic program slicing analyzes dependencies within a particular program execution (Korel & Laski, 1988). RELOOP uses simpler instrumentation to record values and predicates without constructing a dependency graph. Its contribution is a post-generation route from execution events to progress supervision; potential differences and online memory implement that route using established shaping ingredients.

3 PROGRESS SUPERVISION FROM REWARD PROGRAM EXECUTION

3.1 PROBLEM SETTING AND EXECUTION TRACE

Let p be an LLM-generated reward program held unchanged during policy training. It maps environment context after the transition ξ_{t+1} and action a_t to the scalar reward $r_t^p = p(\xi_{t+1}, a_t)$. The policy retains its usual environment observations; RELOOP changes how its training reward is computed. Instrumenting the program call exposes an ordered sequence of runtime events,

$$E_t = (e_i)_{i=1}^{m_t}, \qquad e_i = (n_i, v_i, c_i), \tag{1}$$

where n_i is the recorded name, v_i a finite scalar summary, and c_i the context, such as assignment, condition, or return. Arrays with one element retain their scalar value; larger numeric arrays are summarized by the norm of their finite entries. Events without a finite numeric summary are omitted from the trace. Recorder calls return their inputs, preserving the recorded expressions on the supported Python subset. The supplement specifies the transformation scope, filtering rules, and tests of output equivalence.

3.2 TASK INFORMATION BEYOND THE RETURNED SCALAR

Two program executions can return equal rewards while containing different intermediate values relevant to the task. For example, a weighted reward can offset better gripper proximity against worse object placement. A representation built only from that scalar merges these cases, while a representation retaining both distances can distinguish them. This concerns information available to reward shaping; the policy's environment observations may already distinguish the underlying states.

Table 1 shows the PickCube intermediates used by our grounding rules. The program computes gripper proximity, grasp status, and distance from object to goal before combining them into its reward. RELOOP records these existing computations and maps them to a progress representation. Whether that representation improves learning is tested by the controls in Section 4.

Table 1: **Grounding examples from the PickCube reward program.** Distance and grasp variables map to semantic channels; `reward` contributes a bounded base term to the progress potential.

Recorded source name	Channel	Interpretation
`dist_gripper_cube`	reach	distance from gripper to cube
`grasping_cube`	interact	whether the gripper grasps the cube
`dist_cube_goal`	goal	distance from cube to target
`reward`	base	bounded source reward term

3.3 FROM EVENTS TO A PROGRESS STATE

A deterministic lexical map $g(n)$ assigns event names to four semantic channels, a separate base channel, or no channel. The semantic set is $\mathcal{K} = \{\text{reach}, \text{interact}, \text{goal}, \text{stable}\}$. The transform $\psi(n, v, k)$ depends on the event name and maps retained values to $[0, 1]$ using the rules in the supplement. Smaller distances and errors receive larger scores, Boolean values remain binary, and other values are bounded by monotone transforms. For each observed semantic channel, we retain the largest normalized value,

$$x_{k,t} = \max_{e_i \in E_t : g(n_i) = k} \psi(n_i, v_i, k), \tag{2}$$

with missing channels marked by $\perp$ and excluded from the subsequent average. This aggregation uses the strongest available signal within each lexical category.

The default order is reach $\prec$ interact $\prec$ goal $\prec$ stable, with later channels softly gated by earlier observed channels. Let $A_t = \{k \in \mathcal{K} : x_{k,t} \neq \perp\}$ denote the active set, and let σ be the logistic sigmoid. The gates and structured potential are

$$\begin{aligned} G_{k,t} &= \prod_{j \in A_t : j \prec k} \sigma\left(\frac{x_{j,t} - \tau}{T}\right), \\ \phi_t^{\text{str}} &= \begin{cases} |A_t|^{-1} \sum_{k \in A_t} G_{k,t} x_{k,t}, & |A_t| > 0, \\ 0, & |A_t| = 0. \end{cases} \end{aligned} \tag{3}$$

Here τ sets the gate threshold and T controls its smoothness. The base value b_t is the maximum normalized base event, with a bounded transform of r_t^p used when none survives. The progress potential and discrete trace state are

$$\Phi_t = (1 - \lambda_p)\phi_t^{\text{str}} + \lambda_p b_t, \qquad z_t = Q_B(x_t), \tag{4}$$

where Q_B divides each observed channel into B bins and assigns a separate code to missing values. Changes in Φ_t are the most direct training signal from program execution: equal scalar returns can still yield different potential changes when their intermediate values differ. The channels and order provide a shared heuristic for manipulation tasks, whose value is examined using the ungated Flat control.

3.4 Online progress memory and reward

Each environment worker maintains visits to trace states $N(z)$, transitions $C(z, z')$, success counts $S(z)$, and the largest potential reached in the current episode.

The frontier bonus rewards an increase beyond the previous maximum M_{t-1} by a margin δ,

$$F_t = \frac{\mathbb{I}[\Phi_t > M_{t-1} + \delta]}{\sqrt{N(z_t) + 1}}, \qquad M_t = \max(M_{t-1}, \Phi_t). \tag{5}$$

Visit counts reduce the bonus for trace states encountered frequently across episodes. Success counts record visits at which the success flag provided by the environment is true. At episode boundaries, transition frequencies $\widehat{P}(z' \mid z)$ and success occupancy $\hat{o}(z) = S(z)/(N(z) + \epsilon)$ define a score updated by synchronous backups,

$$V(z) \leftarrow \mathrm{clip}\left(\hat{o}(z) + \gamma_V \sum_{z'} \widehat{P}(z' \mid z) V(z'), -1, 1\right). \tag{6}$$

The transition term is zero for states with no observed outgoing edges. Repeated backups propagate observed success occupancy to preceding trace states. The resulting V is a shaping score, without an interpretation as a calibrated success probability. Episode initialization sets $\Phi_{-1} = 0$, $M_{-1} = -\infty$, and the sentinel value $V(z_{-1}) = 0$.

The complete reward is

$$r_t^{\text{ReLoop}} = \alpha r_t^p + \lambda(\Phi_t - \Phi_{t-1}) + \eta F_t + \beta(\gamma_V V(z_t) - V(z_{t-1})) + \kappa \mathbb{I}[\text{success}_t]. \tag{7}$$

The coefficients are shared across tasks, with $\alpha = 0.30$, $\lambda = 3.0$, $\eta = 0.10$, $\beta = 0.20$, and $\kappa = 1.0$. The last term adds a bonus on each step marked successful by the environment. The potential difference provides a training signal from program execution; frontier expansion and discounted success occupancy additionally use online memory. Because this difference lacks the policy discount and memory changes during training, the complete reward is evaluated empirically without a claim of policy invariance.

3.5 Online update and computational footprint

Algorithm 1 computes each reward from the previous snapshot, the episode maximum M_{t-1}, and worker-local N and V. It then records the current visit, transition, and success flag before updating V at episode end. The previous snapshot and episode maximum reset between episodes; N, C, S, and V persist. The supplement gives the complete memory lifecycle and implementation settings.

Algorithm 1 ReLoop reward update for one environment worker. The current reward is computed before this transition is recorded.

Require: Fixed program p, transition context (ξ_{t+1}, a_t), environment success and episode-end flags, and worker-local memory $(\Phi_{t-1}, z_{t-1}, M_{t-1}, N, C, S, V)$
Ensure: Shaped scalar reward r_t^{ReLoop}

```
Run the instrumented p once to obtain (r_t^p, E_t).
Ground (E_t, r_t^p) into (x_t, b_t, Φ_t, z_t).
Compute F_t by equation 5 and r_t^ReLoop by equation 7 using pre-update memory.
N(z_t) ← N(z_t) + 1; S(z_t) ← S(z_t) + 𝕀[success_t].
if a previous trace state exists in this episode then
    C(z_{t-1}, z_t) ← C(z_{t-1}, z_t) + 1.
end if
M_t ← max(M_{t-1}, Φ_t); store (Φ_t, z_t) as the previous snapshot.
if the episode ends then
    Apply 12 synchronous backups to V via equation 6; reset the snapshot and M.
end if
return r_t^ReLoop to PPO or SAC.
```

Grounding retains at most 512 events per call. Sparse visit and transition tables make episode-end backups depend on observed states and edges; the supplement reports throughput.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate LiftCube and PickCube from ManiSkill2 (Gu et al., 2023), and six Meta-World tasks (Yu et al., 2020) covering contact, articulated objects, and sweeping (Figure 2). ManiSkill2 tasks use PPO (Schulman et al., 2017) for 2M (LiftCube) or 4M (PickCube) environment steps. Meta-World tasks use SAC (Haarnoja et al., 2018) for 1M steps per run. Each combination of task and method uses training seeds $\{0, 1, 2\}$ and fixed evaluation seed schedules.

Programs and comparisons. One archived zero-shot reward program in Text2Reward format is used for each task, with identical source across the reported conditions. The archive preserves program snapshots but lacks the original generation prompts, API responses, and model metadata. The comparison therefore evaluates the training interface after generation; it does not reproduce the full Text2Reward pipeline. **FrozenProgram** directly optimizes the program reward in the shared training harness. **Matched** uses instrumentation, the same 0.30 program scale and $+1$ success bonus as RELOOP, and sets $\lambda = \eta = \beta = 0$. **ReLoop** uses Equation 7, with wrapper coefficients, lexical rules, and discretization fixed across tasks and seeds.

Two diagnostic controls were first evaluated on PickCube, DoorUnlock, and HandlePressSide, chosen using pilot outcomes to cover different response profiles. **ReturnOnly** retains online memory and shaping but builds its state from the scalar program return; it was subsequently extended to all eight tasks. **Flat** retains the trace channels and memory but removes ordered prefix gating on the original three tasks. A further **PotentialOnly** diagnostic retains the term based on the structured potential difference but sets $\eta = \beta = 0$ on PickCube, DoorUnlock, and SweepInto. All diagnostic conditions use three seeds per task. The supplement records their protocols, run cohorts, and complete results.

Metrics. *Final success* is the final policy's success rate over 100 episodes with fixed seeds. For curve metrics, we use checkpoint rates from ten episodes before the training budget and append the final rate from 100 episodes at the budget. *Tail success* averages this curve over its sampled points in the final 20% of training. *Normalized area under the learning curve (AUC)* integrates the unsmoothed curve after prepending $(0, 0)$ and divides by the training budget. We average seeds within each task and then weight tasks equally. The supplement reports variability across independently trained policies and complete results for each seed. ManiSkill2 uses 100-step training truncations and a 200-step evaluation horizon. Paired 100-step endpoint reevaluation preserves the interpretation of the matched comparison, with cell means changing by at most 2.3 percentage points.

4.2 LEARNING FROM THE GENERATED REWARD PROGRAMS

RELOOP increased mean final success across tasks from 70.8% to 87.8% relative to direct optimization of the same programs (Table 2; Figure 3a). Tail success and normalized AUC increased by 13.0 and 7.0 percentage points, respectively. Notable gains include DrawerOpen, from 33.3% to 100.0%, and PickCube, from 51.3% to 88.7% final success. This comparison measures the complete training interface, including reward rescaling and the success bonus; the matched control isolates the added trace-derived terms more closely.

Table 2: **Results across eight tasks.** Entries average three training seeds within each task, then weight tasks equally. Success rates and normalized AUC are shown on a 0–100 scale; parentheses show differences in percentage points from FrozenProgram. Differences are computed before rounding. SweepInto is sensitive to run batch; the supplement gives the complete provenance.

Method	Final success ↑	Tail success ↑	Normalized AUC ↑
FrozenProgram	70.8	72.1	50.9
Matched	80.8 (+10.1)	79.1 (+7.0)	55.1 (+4.2)
RELOOP	**87.8** (+17.1)	**85.1** (+13.0)	**57.9** (+7.0)

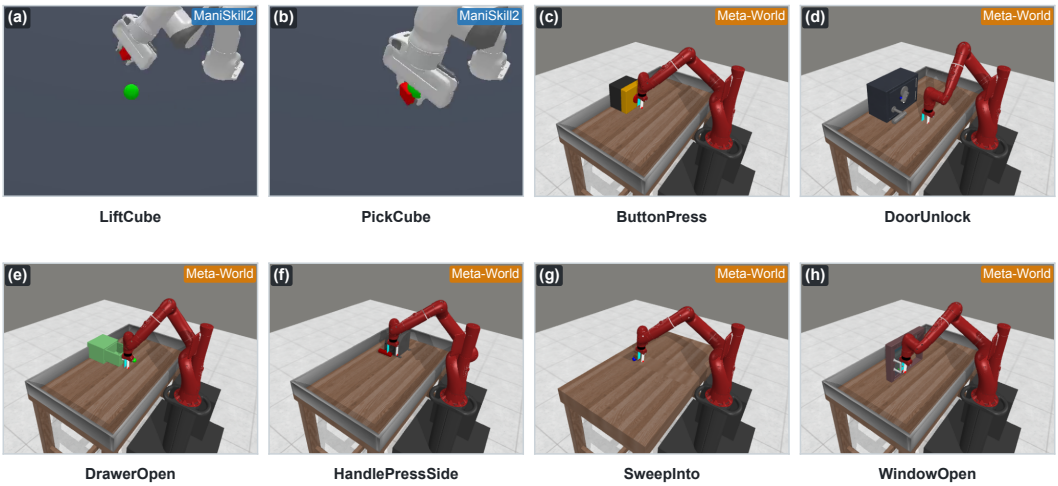


Figure 2: **Manipulation task suite.** LiftCube and PickCube use a Franka arm in ManiSkill2 (blue tags). ButtonPress, DoorUnlock, DrawerOpen, HandlePressSide, SweepInto, and WindowOpen use a Sawyer arm in Meta-World (orange tags). Images illustrate task geometry and robot morphology; the supplement records their rollout provenance.

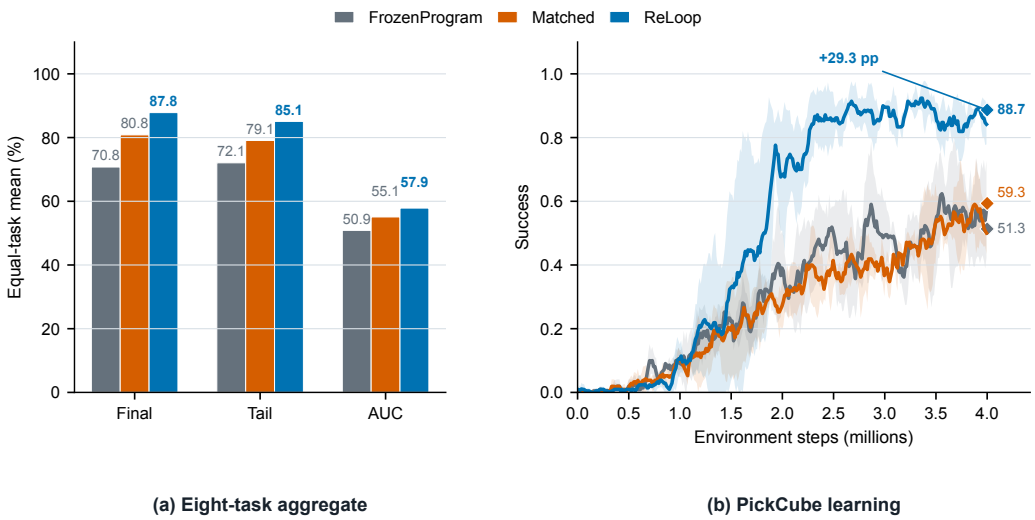


Figure 3: **Aggregate performance and PickCube learning.** (a) Final success, tail success, and normalized AUC across all eight tasks. Each task first averages three training seeds, after which the eight task means receive equal weight. (b) PickCube-only checkpoint success, rather than an average over tasks, smoothed by centered moving averages over seven checkpoints; bands show ± 1 sample standard deviation across three smoothed seed curves. Checkpoints evaluate ten episodes. Diamonds at 4M steps show the separate 100-episode final evaluation, where RELOOP exceeds Matched by 29.3 percentage points. Table metrics use the unsmoothed checkpoint rates and the separate final evaluation. Complete task- and seed-level results appear in the supplement.

4.3 GAINS AFTER MATCHING SCALE AND SUCCESS FEEDBACK

After matching reward scale and success feedback, RELOOP retained average gains of 7.0 points in final success and 2.8 AUC points (Figure 3a). On PickCube, final success reached 88.7% compared with 59.3% for Matched, and normalized AUC increased from 27.0% to 51.5%. The learning curves show this gap developing during training (Figure 3b). Excluding PickCube leaves a +3.8-point final-

success gain; the aggregate AUC gain is concentrated on PickCube. Complete task-level outcomes and sensitivity to task composition are reported in the supplement.

4.4 Structured intermediates and history of scalar returns

On the original three diagnostic tasks, ReLoop exceeded ReturnOnly by 8.9 points in final success and 6.9 AUC points on average. After extending ReturnOnly to the other five tasks, the descriptive gaps across eight tasks were 4.8 and 4.9 points, respectively. These comparisons favor the implemented structured interface. ReturnOnly changes both the information source and the geometry of the derived state, so the contrast does not isolate information content. The extension also includes comparisons across batches; task-level results and run cohorts appear in the supplement.

4.5 Potential change as a component diagnostic

PotentialOnly retains the potential difference derived from program execution while removing frontier and success-occupancy shaping. Its final success exceeded Matched on all three tested tasks and was comparable to or higher than the complete ReLoop reward (Table 3). On PickCube, its 89.3% endpoint and 51.1% AUC were close to the complete method's 88.7% and 51.5%. These results identify potential change as the best-supported component of the trace-derived reward in these settings; they do not establish an incremental contribution from the two memory terms. The runs were performed in a separate batch, and the supplement reports all metrics for each seed and the outcome-conditioned trace diagnostic.

Table 3: **Diagnostic of potential change.** Task means over three seeds on a 0–100 scale. PotentialOnly keeps $\lambda(\Phi_t - \Phi_{t-1})$ and sets $\eta = \beta = 0$; Matched sets all three coefficients associated with the trace terms to zero. Full denotes the complete ReLoop reward. The PotentialOnly runs were trained in a separate batch.

	Final success ↑			Normalized AUC ↑		
Task	Matched	Full	PotentialOnly	Matched	Full	PotentialOnly
PickCube	59.3	88.7	89.3	27.0	51.5	51.1
DoorUnlock	41.0	56.7	59.3	33.8	31.3	32.5
SweepInto	50.7	59.0	72.7	17.8	23.0	30.6

5 Discussion and Limitations

The matched comparison shows that shaping derived from execution can improve learning after holding program scale and environment success feedback fixed. On the three diagnostic tasks, PotentialOnly retains the endpoint gain over Matched after removing the frontier bonus and success memory. The ReturnOnly comparison across eight tasks favors the structured interface on average, but it also changes the geometry of the shaping state. Together, these results support reuse of program intermediates, with the strongest component evidence for the signal based on potential change.

Relationship to reward generation. Methods that generate dedicated progress functions use a separate source of task information (Sarukkai et al., 2024; Zhang et al., 2025c). ReLoop instead reuses computations in the supplied reward program, so it can follow reward generation, search, or refinement without another generation step. Its usefulness consequently depends on the intermediates that program exposes. The present evaluation concerns this downstream training interface; multiple generators and independently sampled programs would test its sensitivity to the supplied code.

Grounding and attribution. The lexical rules depend on informative identifiers and can classify parameters or action regularizers as semantic channels. Our audit of the source code documents this behavior; policy robustness to renaming and equivalent program forms remains untested. Filtering by program dependency could test whether excluding irrelevant computations improves grounding.

Flat provides no evidence that fixed channel ordering improves performance, while PotentialOnly removes frontier and success-occupancy terms jointly and cannot assign a separate effect to either term.

Evaluation scope. The study uses eight simulated tasks, one program per task, and three training seeds per condition. Matched gains are heterogeneous, and several tasks approach ceiling success. ReturnOnly covers all eight tasks, whereas Flat and PotentialOnly are targeted diagnostics; some comparisons use separate execution batches, as recorded in the supplement. The present results apply to the evaluated programs. Multiple independently generated programs and comparisons under a shared budget with generated progress functions or reward machines would test broader generality and relative performance.

6 Conclusion

ReLoop converts computations already made by LLM-generated reward programs into online progress supervision without regenerating code. Across the evaluated programs, this interface improves mean success across tasks over direct optimization and retains an endpoint gain after matching reward scale and success feedback. A diagnostic on three tasks found that potential change alone retained an endpoint advantage over Matched, while the comparison with ReturnOnly across eight tasks favors the structured interface on average. The results motivate studying which program intermediates remain useful across generated programs and task families.

AI use statement

Generative AI tools were used to support literature retrieval, provide feedback on experimental design, discuss the interpretation of descriptive results, assist with software implementation, code organization and testing, and draft or revise manuscript text and figures. The authors verified all AI-assisted work, made the scientific decisions, and take full responsibility for the final submission.

Ethics statement

The experiments use simulated manipulation tasks and do not establish safe behavior on physical robots. Deployment on physical robots would require validation of the reward specification, observation pipeline, and resulting policy under operating conditions.

Reproducibility statement

The supplement documents the study design, task and seed matrix, evaluation protocol, optimizer settings, instrumentation, and grounding rules. It reports task-level outcomes, seed-level endpoint results, run-batch provenance, sensitivity analyses, and the eight frozen reward programs used in the experiments.

References

Jose A. Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. RUDDER: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/16105fb9cc614fc29e1bda00dab60d41-Abstract.html.

Tewodros W. Ayalew, Xiao Zhang, Kevin Yuanbo Wu, Tianchong Jiang, Michael Maire, and Matthew R. Walter. PROGRESSOR: A perceptually guided reward estimator with self-supervised online refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10297–10306, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Ayalew_PROGRESSOR_A_Perceptually_Guided_Reward_Estimator_with_Self-Supervised_Online_Refinement_ICCV_2025_paper.html.

Luigi B. Berducci, Edgar A. Aguilar, Dejan Ničković, and Radu Grosu. HPRS: Hierarchical potential-based reward shaping from task specifications. *Frontiers in Robotics and AI*, 11: 1444188, 2025. doi: 10.3389/frobt.2024.1444188. URL https://doi.org/10.3389/frobt.2024.1444188.

Yanbin Chang, Junfan Lin, Jie Jiang, Runhao Zeng, Changxin Huang, and Jianqiang Li. Master skill learning with policy-grounded synergy of LLM-based reward shaping and exploring. In *International Conference on Learning Representations*, pp. 81832–81866, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/84706cdfc192cd0351daf48f379847e6-Abstract-Conference.html.

Letian Chen, Nina Marie Moorman, and Matthew Craig Gombolay. ELEMENTAL: Interactive learning from demonstrations and vision-language models for reward design in robotics. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 8700–8725, 2025. URL https://proceedings.mlr.press/v267/chen25at.html.

Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Fred Shentu, and Philipp Wu. SARM: Stage-aware reward modeling for long horizon robot manipulation. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1c92edb990a05f2269f0cc3afbb4c952-Abstract-Conference.html.

Roger Creus Castanyer, Faisal Mohamed, Pablo Samuel Castro, Cyrus Neary, and Glen Berseth. ARM-FM: Automated reward machines via foundation models for compositional reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=OBpQdCWLfd.

Sam Devlin and Daniel Kudenko. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pp. 433–440, 2012. URL https://www.ifaamas.org/Proceedings/aamas2012/papers/2C_3.pdf.

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. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 8657–8677, 2023. URL https://proceedings.mlr.press/v202/du23f.html.

Daniel Furelos-Blanco, Mark Law, Anders Jonsson, Krysia Broda, and Alessandra Russo. Hierarchies of reward machines. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10494–10541, 2023. URL https://proceedings.mlr.press/v202/furelos-blanco23a.html.

Ning Gao, Xiuhui Zhang, Xingyu Jiang, Mukang You, Mohan Zhang, and Yue Deng. RF-Agent: Automated reward function design via language agent tree search. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-5742. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/fb9f53edbfd80b3a543f7963b63363ff-Abstract-Conference.html.

Marek Grześ. Reward shaping in episodic reinforcement learning. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems*, pp. 565–573, 2017. URL https://www.ifaamas.org/Proceedings/aamas2017/pdfs/p565.pdf.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. ManiSkill2: A unified benchmark for generalizable manipulation skills. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=b_CQDy9vrD1.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th*

International Conference on Machine Learning, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Kun Huang, Edward S. Hu, and Dinesh Jayaraman. Training robots to evaluate robots: Example-based interactive reward functions for policy learning. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 11–21, 2023. URL https://proceedings.mlr.press/v205/huang23a.html.

Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2107–2116, 2018. URL https://proceedings.mlr.press/v80/icarte18a.html.

Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, 2022. doi: 10.1613/jair.1.12440. URL https://jair.org/index.php/jair/article/view/12440.

Akshat Singh Jaswal, Ashish Baghel, and Paras Chopra. Discovering reinforcement learning interfaces with large language models. *arXiv preprint arXiv:2605.03408*, 2026. URL https://arxiv.org/abs/2605.03408.

Bogdan Korel and Janusz Laski. Dynamic program slicing. *Information Processing Letters*, 29(3): 155–163, 1988. doi: 10.1016/0020-0190(88)90054-3. URL https://www.sciencedirect.com/science/article/pii/0020019088900543.

Minjae Kwon, Ingy ElSayed-Aly, and Lu Feng. Adaptive reward design for reinforcement learning. In *Proceedings of the 41st Conference on Uncertainty in Artificial Intelligence*, volume 286 of *Proceedings of Machine Learning Research*, pp. 2472–2485, 2025. URL https://proceedings.mlr.press/v286/kwon25a.html.

Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. RoboReward: General-purpose vision-language reward models for robotics. *arXiv preprint arXiv:2601.00675*, 2026. URL https://arxiv.org/abs/2601.00675.

Andrew C. Li, Zizhao Chen, Toryn Q. Klassen, Pashootan Vaezipoor, Rodrigo Toro Icarte, and Sheila A. McIlraith. Reward machines for deep RL in noisy and uncertain environments. In *Advances in Neural Information Processing Systems*, volume 37, pp. 110341–110368, 2024. doi: 10.52202/079017-3502. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/c71769e2715835d37c3e25cc1173bd62-Abstract-Conference.html.

Pengyi Li, Jianye Hao, Hongyao Tang, Yifu Yuan, Jinbin Qiao, Zibin Dong, and Yan Zheng. R*: Efficient reward design via reward structure evolution and parameter alignment optimization with large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 34509–34527, 2025a. URL https://proceedings.mlr.press/v267/li25v.html.

Pengyi Li, Hongyao Tang, Jinbin Qiao, Yan Zheng, and Jianye Hao. LaRes: Evolutionary reinforcement learning with LLM-based adaptive reward search. In *Advances in Neural Information Processing Systems*, volume 38, 2025b. doi: 10.52202/085713-0786. URL https://papers.neurips.cc/paper_files/paper/2025/hash/21b5d3a17aa5525f30bfd2bc59ac3a48-Abstract-Conference.html.

Xiaofeng Lin and Xuezhou Zhang. Efficient reinforcement learning in probabilistic reward machines. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(18):18728–18736, 2025. doi: 10.1609/aaai.v39i18.34061. URL https://ojs.aaai.org/index.php/AAAI/article/view/34061.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/70c26937fbf3d4600b69a129031b66ec-Abstract-Conference.html.

Parsa Mahmoudieh, Deepak Pathak, and Trevor Darrell. Zero-shot reward specification via grounded natural language. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 14743–14752, 2022. URL https://proceedings.mlr.press/v162/mahmoudieh22a.html.

Tongzhou Mu, Minghua Liu, and Hao Su. DrS: Learning reusable dense rewards for multi-stage tasks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/22563ee9860728cac644b8cef32431cf-Abstract-Conference.html.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, pp. 278–287, 1999. URL https://ai.stanford.edu/~ang/papers/shaping-icml99.pdf.

Georg Ostrovski, Marc G. Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2721–2730, 2017. URL https://proceedings.mlr.press/v70/ostrovski17a.html.

Norman Di Palo, Leonard Hasenclever, Jan Humplik, and Arunkumar Byravan. Diffusion augmented agents: A framework for efficient exploration and transfer learning. In *Proceedings of The 3rd Conference on Lifelong Learning Agents*, volume 274 of *Proceedings of Machine Learning Research*, pp. 268–284, 2025. URL https://proceedings.mlr.press/v274/palo25a.html.

Vishnu Sarukkai, Brennan Shacklett, Zander Majercik, Kush Bhatia, Christopher Ré, and Kayvon Fatahalian. Automated rewards via LLM-generated progress functions. *arXiv preprint arXiv:2410.09187*, 2024. URL https://arxiv.org/abs/2410.09187.

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

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/98711dea460bdefe0e651ca23ec98ba2-Abstract-Conference.html.

Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Cheng Chi, Yuheng Ji, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, Shaoxuan Xie, Guocai Yao, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. General process reward modeling for robotic reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22412–22422, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Tan_General_Process_Reward_Modeling_for_Robotic_Reinforcement_Learning_CVPR_2026_paper.html.

Efstratios Triantafyllidis, Félix Christianos, and Zhibo Li. Intrinsic language-guided exploration for complex long-horizon robotic manipulation tasks. In *2024 IEEE International Conference on Robotics and Automation*, pp. 7493–7500, 2024. doi: 10.1109/ICRA57147.2024.10611483. URL https://discovery.ucl.ac.uk/id/eprint/10197138/.

Christos K. Verginis, Cevahir Koprulu, Sandeep Chinchali, and Ufuk Topcu. Joint learning of reward machines and policies in environments with partially known semantics. *Artificial Intelligence*,

333:104146, 2024. doi: 10.1016/j.artint.2024.104146. URL https://doi.org/10.1016/j.artint.2024.104146.

Eric Wiewiora. Potential-based shaping and Q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19:205–208, 2003. doi: 10.1613/jair.1190. URL https://doi.org/10.1613/jair.1190.

Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2Reward: Reward shaping with language models for reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/9941833e8327910ef25daeb9005e4748-Abstract-Conference.html.

Jingyun Yang, Max Sobol Mark, Brandon Vu, Archit Sharma, Jeannette Bohg, and Chelsea Finn. Robot fine-tuning made easy: Pre-training rewards and policies for autonomous real-world reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation*, pp. 4804–4811, 2024. doi: 10.1109/ICRA57147.2024.10610421. URL https://ieeexplore.ieee.org/document/10610421/.

Zhaojing Yang, Miru Jun, Jeremy Tien, Stuart Russell, Anca Dragan, and Erdem Biyik. Trajectory improvement and reward learning from comparative language feedback. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 5389–5404, 2025. URL https://proceedings.mlr.press/v270/yang25e.html.

Xin Ye, Feng Tao, Abhirup Mallik, Burhaneddin Yaman, and Liu Ren. LORD: Large models based opposite reward design for autonomous driving. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 5072–5081, 2025. URL https://openaccess.thecvf.com/content/WACV2025/html/Ye_LORD_Large_Models_Based_Opposite_Reward_Design_for_Autonomous_Driving_WACV_2025_paper.html.

Kelin Yu, Sheng Zhang, Harshit Soora, Furong Huang, Heng Huang, Pratap Tokekar, and Ruohan Gao. GenFlowRL: Shaping rewards with generative object-centric flow in visual reinforcement learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13183–13192, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Yu_GenFlowRL_Shaping_Rewards_with_Generative_Object-Centric_Flow_in_Visual_Reinforcement_ICCV_2025_paper.html.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100, 2020. URL https://proceedings.mlr.press/v100/yu20a.html.

Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montserrat Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, Brian Ichter, Ted Xiao, Peng Xu, Andy Zeng, Tingnan Zhang, Nicolas Heess, Dorsa Sadigh, Jie Tan, Yuval Tassa, and Fei Xia. Language to rewards for robotic skill synthesis. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 374–404, 2023. URL https://proceedings.mlr.press/v229/yu23a.html.

Yuwei Zeng, Yao Mu, and Lin Shao. Learning reward for robot skills using large language models via self-alignment. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 58366–58386, 2024. URL https://proceedings.mlr.press/v235/zeng24d.html.

Chen Bo Calvin Zhang, Zhang-Wei Hong, Aldo Pacchiano, and Pulkit Agrawal. ORSO: Accelerating reward design via online reward selection and policy optimization. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/6155cfb4e650d41167848fe2ead30ab8-Abstract-Conference.html.

Jiahui Zhang, Yusen Luo, Abrar Anwar, Sumedh Anand Sontakke, Joseph J. Lim, Jesse Thomason, Erdem Biyik, and Jesse Zhang. ReWiND: Language-guided rewards teach robot policies without new demonstrations. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 460–488, 2025b. URL https://proceedings.mlr.press/v305/zhang25a.html.

Xiuhui Zhang, Ning Gao, Xingyu Jiang, Yihui Chen, Yuheng Pan, Mohan Zhang, and Yue Deng. Progress reward model for reinforcement learning via large language models. In *Advances in Neural Information Processing Systems*, volume 38, 2025c. doi: 10.52202/085713-3370. URL https://proceedings.nips.cc/paper_files/paper/2025/hash/91b047c5f5bd41ef56bfaf4ad0bd19e3-Abstract-Conference.html.

Yinuo Zhao, Jiale Yuan, Zhiyuan Xu, Xiaoshuai Hao, Xinyi Zhang, Kun Wu, Zhengping Che, Chi Harold Liu, and Jian Tang. Training-free generation of temporally consistent rewards from VLMs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8133–8143, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Zhao_Training-free_Generation_of_Temporally_Consistent_Rewards_from_VLMs_ICCV_2025_paper.html.